Pós-Graduação em Ciência da Computação

Ricardo Alves da Silva

DETERMINAÇÃO DE RECURSOS MINERAIS COM UTILIZAÇÃO DE OTIMIZAÇÃO E ALGORITMOS DE APRENDIZADO DE MÁQUINA

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
http://cin.ufpe.br/~posgraduacao

Recife
2019

Ricardo Alves da Silva

DETERMINAÇÃO DE RECURSOS MINERAIS COM UTILIZAÇÃO DE OTIMIZAÇÃO E ALGORITMOS DE APRENDIZADO DE MÁQUINA

Trabalho apresentado ao Programa de Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Doutor em Ciência da Computação.

Área de Concentração: inteligência computacional
Orientador: Ricardo Martins de Abreu Silva
Coorientador: Rodrigo Gabriel Ferreira Soares

Recife
2019

Catalogação na fonte
Bibliotecária Monick Raquel Silvestre da S. Portes, CRB4-1217

S586d Silva, Ricardo Alves da
Determinação de recursos minerais com utilização de otimização e algoritmos de aprendizado de máquina / Ricardo Alves da Silva. – 2019.
138 f.: il., fig., tab.

Orientador: Ricardo Martins de Abreu Silva.
Tese (Doutorado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2019.
Inclui referências e apêndice.

1. Inteligência computacional. 2. Aprendizagem de máquina. I. Silva, Ricardo Martins de Abreu (orientador). II. Título.

006.3 CDD (23. ed.) UFPE- MEI 2019-110

Ricardo Alves da Silva

Determinação de Recursos Minerais com Utilização de Otimização e Algoritmos de Aprendizagem de Máquina

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Doutor em Ciência da Computação.

Aprovado em: 22/02/2019.

_____

Orientador: Prof. Dr. Ricardo Martins de Abreu Silva

BANCA EXAMINADORA

_____

Prof. Dr. Silvio de Barros Melo
Centro de Informática /UFPE

_____

Prof. Dr. Robson do Nascimento Fidalgo
Centro de Informática /UFPE

_____

Prof . Dr. Giorgio Francesco Cesare de Tomi
Departamento Engenharia de Minas e Petróleo / USP

_____

Prof. Dr. Júlio Cesar de Souza
Departamento de Engenharia de Minas / UFPE

_____

Prof. Dr. Cícero Garozzi
Departamento de Estatística e Informática / UFRPE

Decido este trabalho a minha família que foi o porto seguro perante as dificuldades durante este percurso.

AGRADECIMENTOS

Em primeiro lugar quero agradecer a Deus por ter me dado a oportunidade e coragem de realizar este trabalho. Em seguida agradeço aos meus pais, Maria de Fátima Alves da Silva e Miguel Lourenço da Silva (*in memoriam)* pelos valores passados para mim bem como pelo incentivo na busca do conhecimento. Agradeço também ao professor, Ricardo Martins de Abreu Silva, pelo compromisso de me orientar durante essa difícil trajetória, na qual sobreveio diversos desafios, os quais foram superados com muito esforço e dedicação, pelas diversas oportunidades de aprendizado proporcionadas ao longo desta jornada, muitas das quais vão além da pesquisa e contribuem de forma direta para o crescimento pessoal e profissional. Aos professores Rodrigo Soares, Cícero Garrozi, Giorgio de Tomi, Júlio César de Souza, Cleber Zanchetin, Edna Natividade, pela disponibilidade em, de alguma forma, apoiar este trabalho, em participar da banca, alguns desde a qualificação, e pelas contribuições somadas ao trabalho. A Halisso Alberdan, pela convivência no decorrer desta jornada. A todos os supracitados registro o meu agradecimento e o desejo de tê-los como uma fonte de inspiração para avançar em novos desafios. Registram-se os agradecimentos aos vários autores de estudos que, atendendo à solicitação feita via e-mail, prontamente encaminharam seus estudos para que pudessem ser apreciados no contexto de revisão sistemática da literatura realizada. Ao Centro de Informática (CIn/UFPE), pelo espaço físico disponibilizado. À coordenação e professores do Programa de Pós-Graduação em Ciência da Computação do Centro de Informática (CIn) da UFPE, sobretudo, aqueles que tivemos a grata satisfação de tê-los como docentes durante o cumprimento dos créditos do curso. Às secretárias da pós-graduação Socorro e Lilian pela pronta disponibilidade em me atender sempre que precisava, muito obrigado pelas várias solicitações atendidas ao longo do curso. Este trabalho foi apoiado pela Fundação de Amparo à Ciência e Tecnologia do Estado de Pernambuco (FACEPE), por meio do Processo Nº IBPG-0136-1.03/11, à qual agradecemos pelo apoio e ressaltamos a importância que a mesma teve no contexto desta pesquisa.

RESUMO

O investimento em pesquisa mineral é bastante oneroso e muitas vezes arriscado, tanto do ponto de vista da correta identificação da jazida em subsolo quanto do eficiente trabalho de sondagem, que é feito para conhecer a jazida e avaliar a sua viabilidade econômica. Tomando como exemplo o Brasil, um país com um grande potencial extrativo de minérios, como ferro, ouro e níquel, entre outros, surgiu o interesse, nesta pesquisa, de propor a aplicação de metodologias inovadoras e eficazes, na área de Aprendizado de Máquina (AM), que possam dar um tratamento mais sofisticado e de fácil utilização nas análises de estimativa de recursos minerais. A ideia, aqui, é combinar praticidade e eficácia no processo de quantificação dos bens minerais, através de algoritmos de aprendizado de máquina. Trata-se, portanto, de uma forma de tornar os procedimentos de tratamento dos dados de sondagem mais simplificados do ponto de vista operacional. Ademais, é importante lembrar que as operações em pesquisa mineral são desenvolvidas por meio de levantamento das características geológicas do solo ou de perfuração na área de interesse, e, a partir desses dados, é iniciado o estudo de identificação e quantificação de teores de minério na formação geológica mineralizada. Para isso, é realizado um estudo geoestatístico utilizando o algoritmo de krigagem, a fim de quantificar, em termos de teor mineral, os minérios existentes no subsolo em análise. Como o método da krigagem é estatisticamente paramétrico, ele apresenta algumas limitações quanto à modelagem matemática do corpo mineralizado e, em algumas situações, torna-se muito difícil associar um modelo para a análise variográfica do corpo mineral. Na área de Aprendizado de Máquina, existem diversos estudos tratando da utilização de algoritmos que são treinados mediante hipótese não paramétricas, como os *ensembles*, que são o alvo deste estudo. Tais algoritmos não possuem fórmulas predefinidas quanto à análise da formulação do problema. Através deles é possível se chegar a uma melhor percepção da correlação de teores na classificação e na quantificação dos minerais em estudo. Por meio dos métodos não paramétricos, é possível descrever a formação geológica do corpo mineralizado e, consequentemente, realizar uma melhor estimativa dos teores de minério. Dentre esses métodos, vale destacar os *ensembles*, já citados anteriormente, os quais apresentam características peculiares que podem ser bastante úteis no que diz respeito ao desenvolvimento deste trabalho de pesquisa, podendo, enfim, produzir resultados significativos. Assim sendo, resta acrescentar que é necessário que os problemas realcionados a pesquisas em recursos minerais sejam vistos de forma mais objetiva e prática, ou seja, sejam encarados como desafios a serem superados e não como obstáculos intransponíveis.

Palavras-chaves: Aprendizado de Máquina. Krigagem. Algoritmos. Sondagem. Ensembles.

ABSTRACT

Investment in mineral exploration is quite expensive and often risky, both from the point of view of the correct identification of the underground deposit and the efficient survey work, which is done to know the deposit and evaluate its economic viability. As an example, Brazil, a country with a great potential for extraction of ores, such as iron, gold and nickel, among others, the interest in this research was to propose the application of innovative and effective methodologies in the area of Machine Learning (AM), which can provide a more sophisticated and user-friendly treatment in the estimation of mineral resources. The idea here is to combine practicality and effectiveness in the process of quantification of mineral goods, through machine learning algorithms. It is therefore a way of making the procedures for processing survey data simpler from an operational point of view. In addition, it is important to remember that the mineral exploration operations are developed by means of a survey of the geological characteristics of the soil or of drilling in the area of interest, and, from these data, the study of identification and quantification of ore contents in the mineralized geological formation. For this, a geostatistical study is carried out using the kriging algorithm, in order to quantify, in terms of mineral content, the existing minerals in the subsoil under analysis. As the kriging method is statistically parametric, it presents some limitations regarding the mathematical modeling of the mineralized body and, in some situations, it becomes very difficult to associate a model for the variographic analysis of the mineral body. In the Machine Learning area, there are several studies dealing with the use of algorithms that are trained by non-parametric hypotheses, such as the *ensembles*, which are the target of this study. These algorithms do not have predefined formulas for the analysis of the problem formulation. Through them it is possible to get a better perception of the correlation of contents in the classification and quantification of the minerals under study. By means of non-parametric methods, it is possible to describe the geological formation of the mineralized body and, consequently, to make a better estimate of the ore contents. Among these methods, it is worth mentioning the ensembles already mentioned, which have peculiar characteristics that can be very useful in the development of this research work, and can produce significant results. Therefore, it is necessary to add that the problems related to research in mineral resources must be seen in a more objective and practical way, that is, they are seen as challenges to be overcome and not as insurmountable obstacles.

Keywords: Machine Learning. Kriging. Algorithms. Probing. Ensembles.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ABR	AdaBoost
AM	Aprendizado de Máquina
EN	*ElasticNet*
ERR	Extremely Randomized Regressor, ou regressor extremamente randomizado
GBR	Gradient Boosting, ou aumento de gradiente
RFR	Random Forest Regressor, ou regressor de floresta aleatória
RL	Regressão Linear
RM	Regressão Múltipla
SVM	máquina de vetores de suporte - Support Vector Machine

LISTA DE SÍMBOLOS

$\gamma(h)$	Variância em função da separação h
$\mathbf{X}$	Matriz de sondagens
Z(x)	Valor do teor desconhecido x
h	Separação (incremento) entre os teores
C_0	Efeito pepita (nugget effect)
C	Variância máxima
a	Alcançe (range) no variograma
X	Coordenada geográfica na direção x
Y	Coordenada geográfica na direção y
Z	Coordenada geográfica na direção z
$\varphi(x_i)$	Função Kernel para o SVM
H	Espaço de hipóteses (possíveis estimadores)

SUMÁRIO

1 INTRODUÇÃO

O procedimento clássico para a estimativa de recursos minerais consiste em avaliar esses recursos por meio de métodos geométricos, tais como, métodos dos poligonos, área de influência etc. A aplicação mais atual vem por meio da geoestatística e em particular usando o algoritmo de *krigagem*, onde a sua maior aplicabilidade consiste em estudar fenômenos espacialmente correlacionados, ou seja, que apresentam dependências de proximidade. Este fenômeno, do ponto de vista da análise de recursos minerais, é representado, em sua maioria, por um conjunto de pontos em quatro dimensões que são as coordenadas geográficas: *longitude*, *latitude*, *profundidade* e o conteúdo mineral de interesse que neste caso é o *teor* do minério. Portanto, dado que a nossa variável de interesse é o conteúdo mineral, a qual definimos de *teor*, a solução deste problema é encontrar um modelo de regressão que permita mapear a distribuição dos níveis de *teores* espalhados pelo corpo mineralizado.

Para a indústria mineral, a solução dos problemas relativos à correta identificação dos recursos minerais, bem como a quantificação destes, representa um estágio de grande importância para todo o sistema produtivo de mineração, já que todo o investimento econômico é uma função da quantidade e qualidade dos bens minerais contidos na jazida. Portanto, encontrar uma solução que melhore no que diz respeito às avaliações de depósitos minerais é um passo de grande relevância para o campo da indústria mineral.

O estudo geoestatístico tem como ponto de partida um conjunto de observações que constituem uma amostra, entende-se por amostra um testemunho contendo o nível de pureza do bem mineral de interesse e isto é obtido por meio de uma sondagem realizada no terreno que contém a substância mineral de interesse. Através dessas sondagens é feito um levantamento tipológico do solo e também do maciço rochoso, o qual apresenta as características morfológicas do corpo mineralizado e permite avaliar a natureza qualitativa do bem mineral analisado. Essas informações são utilizadas para caracterizar as propriedades do fenômeno espacial em estudo. De fato, o fenômeno espacial desconhecido representa a população da qual uma amostra foi extraída. O procedimento para efetuar uma estimativa inicial do conteúdo mineralógico geoestatisticamente é estabelecido através do método da krigagem, o qual é responsável por apresentar a quantidade e localização do bem mineral de interesse ao longo da jazida. Em outras palavras, o objetivo consiste em representar as regiões nas quais há maior continuidade do bem mineral de interesse e isso serve para ajustar um modelo geológico de teores confiável do depósito mineral e subsidiar as operações de extração do mineral desejado.

Depois de vários estudos acerca de estimativa de recursos minerais é possível ver que estimar o corpo de minério por meio do algoritmo de krigagem, a metodologia geoestatística, implica descobrir o comportamento da distribuição espacial de teores no subsolo por

meio de funções predeterminadas que podem ser em muitos casos uma gaussiana, uma exponencial e uma cúbica.

Portanto, dentro do contexto da estimativa de recursos minerais, há um campo aberto a ser estudado e melhorado do ponto de vista operacional, ou seja, o tratamento dado a esse conjunto de informações representados pelas sondegens pode ser melhorado no que concerne a metodologia aplicada para o reconhecimento mineralógico da jazida. Então os métodos de aprendizado de máquina se apresentam como uma inovação no campo da estimativa de recursos minerais, onde estes podem melhorar substancialmente e consequentemente acelerar a eficiência de elaboração de modelos geológicos de teores, o que facilita todo o trabalho de planejamento de extração e tratamento do bem mineral até atingir o nível de comercialização. A abordagem de aprendizado de máquina apresenta-se como um método que visa descrever o comportamento de uma determinada grandeza mediante o treinamento do algoritmo e, a partir disso, generalizar um padrão para qualquer instância desconhecida. Em outras palavras, podemos considerar nossos dados como sendo os pontos de sondagens e a grandeza, o teor. Mediante o treinamento do algoritmo por meio desses dados é possível gerar uma função que seja capaz de identificar o teor de minério em qualquer ponto dentro do espaço geométrico em que se encontre a jazida.

A técnica de Aprendizado de Máquina (AM) é bastante interessante visto que existem diversos métodos que possuem características que se adaptam a determinados tipos de dados. Neste caso, estamos interessados em encontrar uma metodologia de AM que seja capaz de se ajustar ao processo de manipulação de dados geometricamente espaçados e correlacionados entre si, com o objetivo de classificar as diferentes regiões do corpo mineralizado com o máximo de precisão. Essa classificação de região é representada pela qualidade ou pureza existente nas diferentes localidades da jazida mineral. A correta classificação é imprescindível para avaliar a viabilidade econômica da jazida. É dentro desse contexto que o desenvolvimento desse trabalho está fundamentado, ou seja, encontrar uma metodologia que seja eficiente e eficaz para a determinação da distribuição espacial de teores dentro do corpo mineralizado.

Os modelos de AM aqui trabalhados são baseados em algoritmos que fazem regressão matemática, os quais têm como objetivo encontrar relações matemáticas, não necessáriamente funções, que consigam descrever o comportamento da variável de interesse da melhor forma possível. Portanto, a análise que é feita neste trabalho contempla uma área da engenharia mineral que é de fundamental importância para o respectivo setor industrial, que é a área de estudo qualitativo e quantitativo dos recursos minerais. Portanto, a proposta desta tese é realizar um estudo dos métodos de regressão em AM que consigam aprender o comportamento da distribuição de teores ao longo de toda a jazida mineral, de maneira que a precisão na informação dada seja a maior possível. Chegando-se a essa meta atingir-se-á um novo patamar no que concerne à estimativa de recursos minerais, contribuindo de forma significativa para os estudos realizados em jazidas, otimizando,

assim, a elaboração dos empreendimentos da indústria de minérios.

1.1 O PROBLEMA

Considerando-se um conjunto limitado de pontos $\mathbf{X} = \{\mathbf{x_i}\}_{i=1}^{n}$, onde $\mathbf{X}$ é uma matriz de sondagem, com n linhas, representada pelos vetores $\mathbf{x}_i = (x_{i1}, x_{i2}, x_{i3}) \in \mathbb{R}^3$, e x_{i1}, x_{i2} e x_{i3} denotam a longitude, latitude e profundidade, respectivamente. Precisamos encontrar uma relação matemática que seja capaz de prever o comportamento da distribuição de teores ao longo de um corpo mineralizado. Com base em tais informações, pode-se chegar à solução do problema encontrando-se a função $\widehat{f}\,(\mathbf{x}) : \mathbb{R}^3 \mapsto \mathbb{R}$, que seja capaz de representar o *teor* do corpo mineralizado ao longo da jazida em conformidade com a equação abaixo:

$$\widehat{f}\,(\mathbf{x}) = \underset{\widehat{f}}{\mathrm{arg}min} \left\{ \sum_{i=1}^{n} \left[\widehat{f}(\mathbf{x_i}) - f(\mathbf{x_i}) \right]^2 \right\} \tag{1.1}$$

Em outras palavras, o problema consiste em tentar encontrar um regressor que permita aproximar a relação 1.1, de forma que a diferença entre a função regredida e a função real, $\widehat{f}(\mathbf{x_i}) - f(\mathbf{x_i})$, seja mínima, isto é, o erro de estimativa seja minimizado, resultando em uma função de regressão cuja proprietade esteja fundamentada na diminuição da variância sem um aumento significativo do viés (*bias*). Essa propriedade da variância minima é própria dos métodos de comitê, visto que eles conseguem combinar diversos regressores de forma que a variância do erro seja atenuada.

Dado um determinado corpo de minério e deseje-se estimar o teor ou pureza de um elemento dentro desse corpo por meio de uma regressão, sendo isso feito a partir de um conjunto finito de pontos, faz-se necessário definir os elementos que compõem o conjunto de treinamento do banco de dados. Esses elementos são formados pelas coordenadas geográficas e pelo *teor* de minério associados a cada ponto do espaço mineralizado. Por exemplo: (x_{i1}, x_{i2}, x_{i3}) representam os elementos da i-ésima linha da matriz $\mathbf{X}$, os quais são respectivamente: *longitude*, *latitude* e *profundidade*. A variável alvo ou independente é representada pelo *teor* ou *pureza* do minério nesse determinado ponto. Todos esses dados serão obtidos por meio de uma sondagem em pontos espaçados da amostra em estudo de forma a mensurar a variação do nível de teor ao longo do sólido em análise. Tendo em vista que se trata de um problema de análise da correlação espacial existente entre os pontos, mediante a distância entre eles, conforme define (GRINGARTEN; DEUTSCH, 2001), entende-se que existe uma forte relação de similaridade entre os teores de acordo com a variação da distância que os separa.

Portanto, com respaldo no contexto supracitado, a proposta desta pesquisa é achar uma solução que consiga identificar a melhor correlação espacial existente entre os teores de minério e quantificar o seu grau de pureza. Para isso, foram avaliadas metodologias de Aprendizado de Máquina, em particular os métodos de comitê (ou *ensembles*), para a predição desses teores em minério de ferro, ouro e níquel.

1.1.1 Jazidas de ferro, níquel e ouro

As jazidas de ferro e níquel possuem características anisotrópicas semelhantes. Elas apresentam baixa descontinuidade, o que significa, do ponto de vista geológico, que elas possuem uma formação espacial mais comportada, possibilitando a aplicação de métodos de aprendizado de máquina que avaliem uma medida de similaridade mais precisa e, a partir disto, consigam identificar as melhores divisões de grupos de teores. Por outro lado ajudam a produzir modelos geológicos de teores mais consonantes com a realidade. Já no que se refere às jazidas de ouro, a descontinuidade da distribuição mineral é mais acentuada e, em alguns casos, apresentam um comportamento bastante anômalo de uma localização para a outra dentro do corpo mineralizado. Trata-se de um fato muito interessante em análise de recursos minerais, uma vez que a aplicação geoestatística clássica encontra algumas dificudades para produzir um modelo geológico de teores mais próximo do padrão apresentado pela natureza.

Com base nas características de continuidade e descontinuidade referentes ao minérios citados foi realizado um estudo mais detalhado de alguns métodos de aprendizado de máquina que possuíssem uma boa eficácia no treinamento dos algoritmos de regressão por meio de medidas de similaridade. Daí, conclui-se que os métodos de comitê possuem um bom desempenho na construção de modelos geológicos de teores para as jazidas minerais supracitadas.

A demanda para a construção de modelos geológicos de teores de minério que representem com maior precisão as jazidas minerais que se encontram no subsolo da terra tem crescido significativamente. E isto se deve ao fato de que as jazidas remanescentes já não são mais fáceis de serem identificadas, o que implica um estudo prévio de pesquisa mineral para obter informações sobre o comportamento espacial dos teores. Esta identificação se dá, basicamente, através de sondagem do terreno, que consiste na realização de furos para quantificar a pureza ou o teor do minério existente em determinada área. Para tanto, são realizadas várias perfurações, cujas profundidades são variáveis de acordo com o tipo de minério que se pretende explorar, bem como a topografia do terreno.

As substâncias minerais relevantes, encontradas no subsolo, estão, na maioria das vezes, associadas a outras substâncias que podem ser de interesse ou não, gerando a necessidade de se estimar não apenas uma variável e sim um conjunto delas. Por exemplo: se há interesse em descobrir o teor de ferro em uma jazida que contenha vanádio e titânio, que também são minerais importantes, faz-se necessário estimar todos esses componentes ao mesmo tempo, visto que eles igualmente representam bens minerais de consumo. Portanto, o trabalho de estimativa, com um número de minerais de interesse elevado, torna a análise, pelo método geoestatístico bem mais difícil. Tendo em mente tal grau de dificuldade para desenvolver um modelo geológico de teores que represente o mais fielmente possível as amostras extraídas do subsolo, propõe-se analisar recursos minerais com o auxílio de algoritmos de aprendizado de máquina, conforme mencionado anteriormente,

visando encontrar o melhor conjunto de regressores computacionais que possam definir o comportamento espacial dos teores minerais avaliados.

O uso dos métodos computacionais apresentados nesta pesquisa visa simplificar o procedimento de estimativa e operar com um número maior de variáveis de interesse (variáveis independentes). Além disso, a estratégia pretende otimizar o procedimento de reconhecimento de padrões minerais das jazidas em estudo.

Os profissionais (engenheiros de minas e geólogos) utilizam, quase que em sua totalidade, as técnicas clássicas de estimativa de recursos minerais, as quais serão mencionadas nesta pesquisa. As técnicas empregadas estão classificadas em determinísticas, que estão fundamentadas apenas na componente geométrica do terreno que é, basicamente, a correlação espacial existente entre eles ou estocásticas, que tem como base fundamental o conceito de aleatoriedade dos dados. Propõe-se, diante desse cenário, e levando-se em conta o contexto apresentado até o momento, uma metodologia que possa representar de forma eficiente e eficaz a distribuição dos teores em uma jazida mineral, e com isto apresentar os modelos geológicos de teores com o uso da krigagem e dos algoritmos de aprendizado de máquina, tendo como meta possibilitar uma análise mais prática na operacionalização dos dados de sondagem.

1.1.2 Questão de Pesquisa

A proposta desta pesquisa é estimular a utilização de procedimentos, mediante a aplicação de aprendizado de máquina, como forma de obter uma maior eficácia na estimativa de recursos minerais. Além disso, o estudo pretente, ainda, apontar caminhos que possibilitem um planejamento mais célere e eficiente em relação ao uso dos recursos minerais disponíveis, visando contribuir de forma bastante significativa para a elaboração de qualquer projeto de mineração. Portanto, dito isto, surge a questão que norteia o presente trabalho: Qual o impacto do uso de algoritmos de AM para a predição da distribuição de teores de minério em uma jazida mineral?

1.2 OBJETIVOS

Tendo como referência a questão de pesquisa definida anteriormente, o trabalho constituise do objetivo geral informado a seguir, bem como os objetivos específicos listados na sequência que cooperam para alcançar tal objetivo.

1.2.1 Objetivo Geral

Elaborar uma metodologia de estimativa de teores em jazidas minerais, por meio do processo de AM, que seja relativamente fácil de ser operacionalizada e que apresente um alto desempenho preditivo.

1.2.2 Objetivos Específicos

1. Desenvolver uma metodologia computacional de precisão que possibilite a obtenção de resultados, de forma eficiente, no tocante à estimativa da distribuição de teores de minérios;

2. Reduzir o erro de estimativa de teores ao longo do corpo mineralizado;

3. Determinar os algoritmos que melhor se apresentem para a estimativa de recursos minerais.

2 ESTADO DA ARTE

Este capítulo corresponde ao referencial teórico do presente trabalho. São abordados conceitos de modelagem geológica com o uso dos algoritmos clássicos de estimativa de recursos bem como os métodos de aprendizado de máquina.

2.1 TRABALHOS RELACIONADOS

A aplicação de algoritmos de AM em recursos minerais é uma aplicação bastante inovadora, uma vez que praticamente todas as publicações concentram-se na utilização de metodologias geoestatísticas as quais vêm sendo aplicadas há décadas pela indústria mineral. Portanto, com o avanço da Inteligência Artificial, foi possível mudar gradativamente a abordagem de solução de um conjunto variado de problemas do mundo real. E dentro deste conjunto de problemas do mundo real insere-se o de estimativa de recursos minerais, o qual é de grande relevância para o setor da indústria mineral.

Os algoritmos de AM têm sido aplicados com bastante êxito em diversos campos da ciência que tratam de problemas de predição ou estimativas em diversos campos da ciência (MENDES-MOREIRA et al., 2012; WU et al., 2008) tais como: Aplicações ecológicas (CRISCI; GHATTAS; PERERA, 2012), genética (LIBBRECHT; NOBLE, 2015), robótica (RANI et al., 2006), séries temporais (KIM, 2003) entre outros.

Dentro da categoria dos algoritmos de AM é importante destacar os *ensembles* porque eles possuem a capacidade de combinar multiplos regressores com o objetivo de reduzir a variância ou viés nos modelos de AM (GEURTS; ERNST; WEHENKEL, 2006). Esta característica é de grande importância para a estimativa de recursos minerais visto que tem-se como meta minimizar a variância do erro de estimativa de teores em uma jazida mineral.

Tendo como base a metodologia de AM fundamentada nos métodos de *ensembles* para a análise de recursos minerais foi feito um estudo da literatura atual para constatar que a nossa abordagem permite avançar o Estado da Arte no que se refere aos estudos de estimativas de recursos minerais.

Dentro do contexto de AM vemos que a aplicação dos métodos baseados em comitê ou *ensemble* não contemplam o campo da indústria mineral no que concerne ao estudo de estimativas de recursos minerais. Portanto, temos contribuído mediante este trabalho para um avanço nas avaliações de recursos minerais. Isto se deve a ótima capacidade de treinamento dos dados e a geração de modelos de AM consistentes com o comportamento das jazidas minerais. Em particular os algoritmos de florestas aleatórias e florestas extra aleatórias.

O geoprocessamento pode ser definido como um conjunto de tecnologias destinadas a coletar e processar informações espaciais para uma finalidade específica. O termo geopro-

cessamento é utilizado para denotar o conjunto de técnicas matemáticas e computacionais para o tratamento da informação geográfica. A aplicação dessa técnica não se limita apenas ao campo da mineração, uma vez que sua aplicação ocorre nas áreas de planejamento urbano, transporte, petróleo e no nosso caso específico na análise de estimativas de recursos minerais. As técnicas utilizadas no geoprocessamento de informação são: sensoriamento remoto, sistema de informação geográfica e geoestatística. A geoestatística tem sua base prática e teórica baseada em vários estudos da ciência do solo.

A estimativa de recursos minerais é essencialmente uma das fases mais importantes das existentes em uma operação mineira, a qual deve ser bem planejada e projetada. A estimativa de um recurso mineral é um problema estatístico e envolve a determinação do valor (ou quantidade) do teor de minério em áreas não amostradas onde estas quantidades são obtidas por meio da interpolação de dados amostrais (geralmente amostras de furos de sondagem) (DUTTA et al., 2010).

Algumas aplicações de Machine Learning para encontrar a melhor função de regressão para estimar dados espacialmente distribuídos mostram várias possibilidades de estimar o teor de minério, uma vez que, a combinação de diversos regressores o torna um método bastante versátil no que concerne a elaboração de modelos de estimativa de recursos minerais.

Por meio dessa perspectiva o uso de métodos de comitê representa uma poderosa metodologia para descobrir o modelo de regressão que melhor modela o comportamento dos teores dos minerais no subsolo. Já que a essência do ensemble é combinar modelos para que o resultado possa ser melhorado.

O procedimento clássico para estimar recursos minerais é baseado em geoestatística, mais especificamente no algoritmo de krigagem. Embora haja várias versões de krigagem, como krigagem lognormal e krigagem indicadora que aplicam certas transformações específicas para capturar relacionamentos não lineares, elas podem não ser eficientes o suficiente para capturar a natureza ampla da não-linearidade espacial, (DUTTA et al., 2010), portanto, esse método executa melhores resultados somente quando um mineral específico está sendo avaliado. Mas na vida real não é muito provável que isso aconteça, já que na maioria das vezes é procurado estimar não apenas um mineral em particular, mas também outros minerais que possam estar associados a ele. Diante deste cenário, a proposta de um algoritmo de Aprendizado de Máquina que possa trabalhar com esses dados de forma linear ou não linear representa uma excelente possibilidade de avançar no campo da mineração no que diz respeito ao estudo da estimativa de recursos minerais.

2.2 AVALIAÇÃO GEOESTATÍSTICA

A Geoestatística é um método relativamente novo de valoração de depósitos minerais desenvolvido nos últimos vinte anos, e foi descrito como uma aplicação da teoria das variáveis regionalizadas ao estudo de volumes mineralizados de rocha e todas as conside-

rações decorrentes disto (Royle, 1977). Os valores dos teores das amostras de sondagem são diretamente influenciados por processos geológicos ao longo do tempo e podem, assim, ser consideradas como variáveis regionalizadas cujos valores desses teores apresenta-se como uma função da sua posição dentro do depósito minério. Qualquer amostra tirada em uma posição $\mathbf{X}$ dentro do depósito terá, portanto, um valor de ensaio F($\mathbf{X}$), onde F($\mathbf{X}$) é uma função de ($\mathbf{X}$).

Contudo, muitos estudos em estimativa de recursos minerais ainda empregam os métodos anteriores ao advento da geoestatística os quais não levam a teoria da variáveis regionalizadas em consideração. Esses métodos fornecem resultados relativamente confiáveis para a estimativa dos teores minérios tendo em vista que praticamente todos esses métodos fundamentam-se na estimativa geométrica para dados espaçados, ou seja, avaliam um ponto desconhecido mediante variações espaciais dos pontos amostrais circunvizinhos.

Segue uma breve descrição dos métodos tradicionais aplicados para estimativas de recursos minerais. Esses métodos são avaliados aqui, apenas para ilustrar as diversas formas de avaliação de recursos minerais, e suas limitações comparadas com o método geoestatístico de análise mineral.

2.3 MÉTODOS GEOMÉTRICOS PARA AVALIAÇÃO RECURSOS

Métodos geométricos possuem uma forma bastante simples de analisar de avaliar os recursos minerais, uma vez que, eles, basicamente, fundamentam-se na geometria Euclidiana. Esses métodos, fundamentalmente, concentram-se nos princípios da interpretação do comportamento de variáveis entre dois pontos adjacentes de amostragem, onde esses pontos determinarão a construção de blocos aos quais serão atribuídos teores para o cálculo das recursos minerais.

2.3.1 Método dos Polígonos

Este método foi bastante aplicado no início das análises de estimativas minerais. Hoje em dia, algumas vezes, ele é utilizado apenas como uma verificação face as estimativas geoestatísticas. Ele é usado para estimar as recursos totais de uma área mineralizada. O método dos polígonos é bastante simples, pois ele consiste em dividir o depósito mineral em uma série de polígonos centrados em amostras individuais pelas bissetrizes perpendiculares das linhas traçadas entre os pontos de amostragem. Cada um dos polígonos é assumido como tendo um valor médio de teor igual ao do valor central da amostra. O teor médio da área é obtido pela ponderação do polígono por sua área de influência. Este pode ser um método bastante trabalhoso, especialmente se as amostras foram retiradas de uma "grade"irregular, o que significa que cada área poligonal deve ser medida individualmente. Os erros desse tipo de estimativa variam de acordo com o tamanho do polígono e, uma vez

que têm-se uma quantidade significativa de áreas poligonais isto aumentará a propagação de erros para a estimativa final do depósito mineral.

2.3.2 Método do Triângulo

Este método é muito semelhante ao método dos poligonos e é muito simples de usar. Para a aplicação desse método basta apenas unir os pontos dos locais onde foi realizada a da amostra para formar os polígonos triangulares e o valor atribuído a cada triângulo é a média ou a média ponderada dos valores em cada vértice do triângulo. Neste caso, os triângulos individuais são avaliados com menos erro do que o método poligonal, uma vez que mais valores amostrais estão envolvidos na estimação. O teor médio da área é calculado da mesma forma que para o método dos poligonos e por esta razão o nível de erro será reduzido se uma amostra é avaliada em triângulos regulares.

2.3.3 Grids Estratificados Aleatórios

Este método pode ser usado quando uma amostragem padrão se aproxima de um *grid* regular, mas não chega a atingi-lo devido variações no mergulho do terreno e nas deflexões de perfuração (Royle, 198). Um *grid* regular é montado nas amostras de tal maneira que cada painel contenha apenas um valor de amostra que tem uma posição aleatória dentro do painel quando comparado com os outros painéis do *grid* e seus valores de amostra. O tamanho do *grid* pode ser obtido dividindo a área total de interesse pelo número de amostras nele contidas a fim de estabelecer a área de um painel. A relação dos lados de cada painel pode ser obtido examinando o padrão de amostragem, com o objetivo de ter apenas um valor de amostra dentro de cada painel do *grid*. Se duas ou mais amostras existem dentro de um painel do *grid*, nesse caso, é necessário retirar a média aritmética do conjunto. Em alguns casos existem painéis que não contêm valores de amostra, mas são rodeados por painéis com amostras. Supõe-se então que estes valores circundantes podem ser projetados no painel e a média aritmética é atribuída ao painel não amostrado.

Este método difere do método poligonal e triangular, uma vez que todos os painéis têm as mesmas dimensões, eliminando assim a necessidade de numerosos cálculos da área total do *grid*. Desta forma o valor do erro é minimizado, pois os valores da amostra são estendidos sobre a mesma área de influência.

2.3.4 Seções e Contornos

As seções e os planos das distribuições dos valores da amostra podem ser desenhadas por meio de intervalos regulares em toda a área de interesse. A amostra e os valores geralmente são contornados à mão ou por computador. Um planímetro é usado para medir a área entre os limites de contorno. O valor é atribuído a esta área é o da média aritmética dos

dois contornos. Este é um método muito artesanal, embora muitas vezes eficaz em casos simples.

2.4 MÉTODOS QUE COMBINAM TÉCNICAS GEOMÉTRICAS E ESTATÍSTICAS

Existem métodos para a estimativas de recursos minerais que combinam as técnicas geométricas com a aplicação da estatística. A seguir serão mostrados os principais métodos que combinam esses dois entes.

2.4.1 Ponderação da Distância

Este método tornou-se popular com a introdução do computador principalmente porque é necessário um grande número de cálculos repetitivos. Um bloco ou painel é atribuído a uma combinação dos valores circundantes, ponderados, por exemplo, pela sua distância ou distância ao quadrado, a partir do centro de bloco. A questão agora surge quanto ao número máximo e mínimo de amostras a serem incluídas no cálculo e a que distância a amostra mais distante pode estar. Geralmente, como o número de valores de amostra aumenta, então sua influência se aproxima de 1/n, onde n é o número de valores da amostra. Este procedimento está bastante associado a experiência do responsável pela análise da estimativa, visto que os ponderadores podem variar de acordo com o tipo de minério, bem como o arranjo espacial que ele se encontra. Portanto, apesar de ser um método aplicável para realizar estimativas, se faz necessário ter bastante conhecimento geológico.

2.4.2 Superfície de Tendência

Superfícies de tendência envolvem a adaptação de uma superfície de quadrados curvilíneos mínimos aos dados da amostra, de forma que as coordenadas geográficas dos valores da amostra podem ser usados para estimar o valor da amostra. Quaisquer tendências dentro dos dados são destacados e modelados por uma simples equação. Esta técnica pode ser facilmente aplicada a depósitos relativamente simples, por exemplo: carvão e bauxita.

2.5 ERROS DE ESTIMATIVA

Todos os métodos mencionados até o momento são capazes de atribuir valores estimados de teor para uma determinada área que se deseja fazer uma análise da ditribuição espacial dos componentes minerais nela existentes. Mas quão precisa será esta estimativa? A maneira mais viável de estabelecer isso seria avaliar, em tempo real, a quantidade de minério extraída, medir o teor resultante, comparando-o com o seu valor estimado. Mas, na prática, não é o que acontece. Portanto, o melhor método de avaliação será aquele que

produza a menor diferença entre o valor real e o calculado, de maneira que sejam minimizados erros no planejamento da extração do bem mineral. Nos casos supramencionados, costuma-se projetar um único valor a uma área, na maioria das vezes, muito grande, o que gera uma somatório de erros para a estimativa final. Não se leva em conta as possibilidades de erros de amostragem e se elas existirem quão grandemente serão ampliadas. Um erro real de qualquer estimativa que é simplesmente a diferença entre os valores reais e os estimados, enquanto que a variância da estimativa é a diferença do valor real e estimado ao quadrado. Quanto menor esse valor, melhor a técnica aplicada para a estimativa. Em linhas gerais os erros de estimativa podem ser atenuados quando possuímos o máximo de informação sobre a reserva mineral estudada, tais como: formação geológica, malha de sondagem bem distribuída, amostragem e classificação dos elementos minerais de forma correta. Com essa prática fica mais fácil minimizar os erros decorrentes da análise realizada pelos métodos supracitados. Mas, na prática, combinar todos esses elementos que minimizam o erro não é tarefa fácil, visto que as imprecisões nas medições bem como os altos valores para a realização dos furos de sondagem sugerem que apliquemos métodos mais sofisticados para, mesmo sabendo desses eventuais erros dos dados amostrais, possamos obter uma estimativa confiável para problema proposto. É dentro desse contexto que surgiu a geoestatística, ou seja, a busca por métodos mais robustos que produzissem resultados confiáveis, mesmo em situações não tão favoráveis.

2.6 O USO DA GEOESTATÍSTICA NA AVALIAÇÃO DE RECURSOS MINERAIS

A geoestatística é um método que foi implementado e aplicado para avaliar os depósitos minerais nos últimos 50 anos, e tem sido descrita como a aplicação da teoria de variáveis regionalizadas ao estudo de volumes mineralizados de rocha e todas as considerações decorrentes disso. A teoria geoestatística foi desenvolvida pelo matemático francês Georges Matheron, o qual tomou como base os dados experimentais obtidos nas minerações de ouro da Africa do Sul onde Daniel G. Krige trabalhou. Antes do advento da geoestatística e do trabalho de Matheron, técnicas geométricas e estatísticas foram usadas com a suposição de que os valores amostrais retirados de um volume mineralizado fossem distribuídos aleatoriamente. No entanto, isso é muito raramente o caso. Os valores das amostras são diretamente influenciados por processos geológicos e podem, portanto, ser considerados como variáveis regionalizadas cujos valores são uma função de sua posição dentro do depósito. Qualquer amostra tomada na posição $\mathbf{X}$ dentro do depósito terá, portanto, um valor de ensaio $F(\mathbf{X})$, onde $F(\mathbf{X})$ é uma função de $(\mathbf{X})$. No entanto, muitas empresas de mineração ainda empregam os métodos tradicionais mais antigos, que não levam isso em consideração. Em alguns casos, eles fornecem resultados confiáveis para seu método particular de operação e, portanto, não têm motivos para mudar para métodos geoestatísticos.

2.7 A CONCEPÇÃO GEOESTATÍSTICA

Nenhum dos métodos mencionados até agora parece ser um método robusto de estimação, pois há diversas maneiras pelas quais os erros podem ser gerados. Parece razoável tentar usar um método o qual é baseado no fato de que os valores das amostras podem ser representados por uma função de sua posições dentro do espaço do depósito analisado. No entanto, o comportamento espacial do valores podem ser complexos, visto que, existem áreas dentro do depósito onde os valores da amostra são predominantemente altos e outras áreas onde os valores são de baixo valor, sugerindo que existe uma probabilidade de que os valores exercem influência um sobre o outro em função da sua localidade. Ainda há casos nos quais não é possível modelar essa função espacial, que são nos casos de jazidas erráticas, ou seja, aquelas que não possuem correlação espacial, o que dificulta bastante realizar uma estimativa eficiente nesses casos. Portanto vemos que o objetivo da geoestatística é modelar uma função que possa definir o comportamento, da variável estudada, distribuído em toda a região de interesse. Para que isso possa ser realizado é imprescindível que os valores da amostra estejam correlacionados entre si por meio de uma função que depende da distância entre os dois valores amostrais. Na maioria dos casos os valores de baixo teor existem dentro de uma quantidade maior de área, enquanto que os valores de alto teor são encontrados em dimensões menores de área, o que corrobora com a concepção da influencia da localidade no teor do minério. Mas, isso não significa, necessariamente, que duas amostras tomadas juntas tem o mesmo valor, pois a estratificação do terreno bem como a distribuição espacial aleatória dos teores, não permitem a igualdade de valores amostrais. Portanto, a variação total entre dois valores de amostra agora pode ser considerada como a soma da variação aleatória local com sua variação espacial.

Os métodos tradicionais não levam esses fatores em conta, mas simplesmente extrapolam os valores da amostra para uma determinada área, independentemente de quaisquer influências espaciais ou estruturais.

2.8 O VARIOGRAMA

A variabilidade entre um conjunto de valores amostrais pode ser medida pela função de variograma. Considere dois valores de amostra, $Z(x)$ e $Z(x+h)$, localizado nas posições x e $x+h$ respectivamente. A diferença de valor é $[Z(x)-Z(x+h)]$. Portanto, considerando essa diferença, podemos definir que o variograma é a variância da diferença entre uma variável x localizada em dois pontos diferentes do espaço. Matematicamente, o variograma é dado pela seguinte expressão: $2\gamma(x, x+h) = E[Z(x) - Z(x+h)]^2$ onde E representa a esperança matemática. Através desse entendimento formamos a base para aplicação do variograma experimental, o qual tem como objetivo definir o comportamento da variável regionalizada no subsolo que em nosso estudo é representada pelos teores de ferro, ouro e

níquel.

O variograma experimental, para uma quantidade n de pares é representado pela equação abaixo:

$$\gamma(h) = \frac{1}{2n}\sum_{i=1}^{n}[Z(x_i) - Z(x_i + h)]^2 \tag{2.1}$$

Portanto, o variograma experimental será produzido por meio da plotagem da variância espacial $\gamma(h)$ contra suas respectivas distâncias h, uma vez que o variograma experimental considera as variações das componetes em todo o espaço mineralizado em análise.

Uma representação típica para um variograma experimental está representada na figura 1.

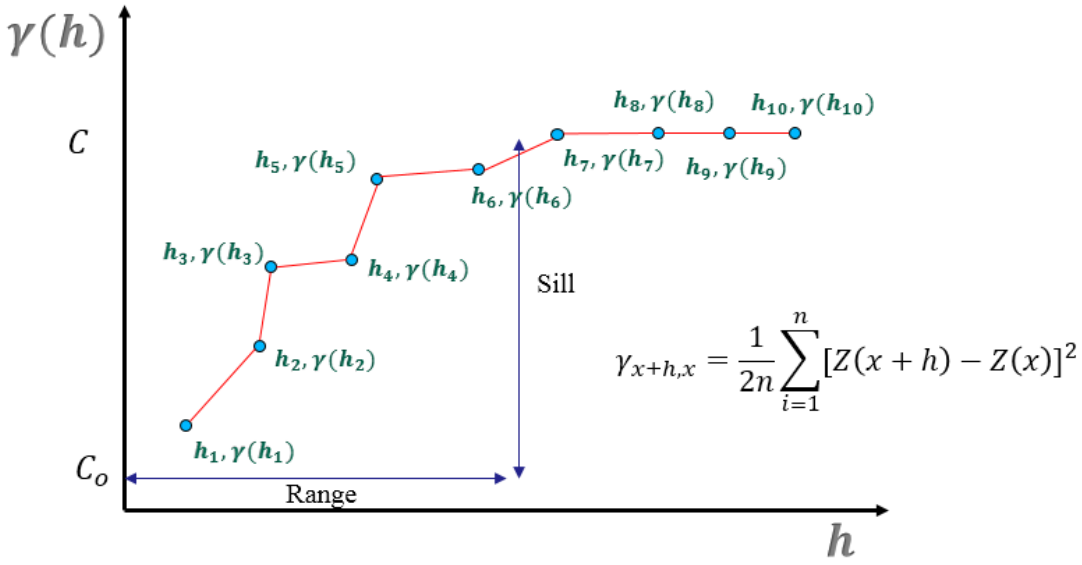

Figura 1 – Ilustração de um variograma experimental

Onde o Range corresponde ao alcançe máximo, ou seja, aquele onde ainda existe correlação entre os teores do minério e o Sill (*patamar*) representa o ponto em que a variância é estabilizada.

2.8.1 Propriedades do Variograma

1. A Variância *Nugget* $C(0)$: Na maioria dos casos, os variogramas experimentais não tendem a zero no desafastamento ou lag $(h) = 0$, mas interceptam o eixo Y em algum valor positivo $\gamma(0) \geq 0$. Esta parte representa o componente aleatório da variação da amostra e é frequentemente o resultado de técnicas analíticas pobres, mineralização errática ou quaisquer estruturas que podem ocorrer a uma distância menor do que aquela do intervalo de desafastamento. A variância *nugget* também recebe o nome de $Efeito\ Pepita$, (JOURNEL; HUIJBREGTS, 1978).

2. A Continuidade (C): A continuidade do variograma é dada pela taxa de aumento da variância com a distância de desafastamento h. Isto dá uma indicação da taxa na qual existe a diminuição da influência de uma amostra sobre a outra ao longo de todo espaço analisado. Quando não há nenhuma continuidade, entende-se que não há correlação espacial entre as amostras, o que significa que a jazida é completamente erratica, em outras palavras, não é possível compor um variograma experimental que represente esse dados.

3. O Alcançe do Variograma (a): Um ponto é alcançado no variograma onde a continuidade é zero, ou seja, a variância é constante, e é nesse ponto que o variogramama atinge um patamar ou $Sill$ e a distância horizontal em que isso ocorre é chamada de alcançe ou $range$. A distância em que isso ocorre dentro do variograma é uma indicação de que além dessa distância os valores de amostra não estão mais relacionados entre si.

2.8.2 Anisotropia

Embora não seja o escopo desta pesquisa discorrer sobre a 'anisotropia', parece adequado tecer alguns breves comentários acerca do tema, visto que existe uma ligação deste com a estimativa de recursos minerais.

Anisotropia, de acordo com o Dicionário inFormal da Língua Portuguesa, é "a característica que uma substância possui em que uma certa propriedade física varia com a direcção". O termo é oriundo das palavras (ani: não, iso: igual, tropia: volta). No tocante à Mineralogia, a referida publicação apresenta a seguinte definição: "Características próprias de certos minerais que se manifestam de modo diferente, conforme as direções cristalográficas consideradas [...]". Portanto, com base neste conceito pode-se afirmar que a anisotropia representa as diferentes variações na continuidade de teores do corpo mineralizado, as quais são observadas mediante as variações nas direções ângulares, ou seja, a variância amostral dos pares de teores é calculada em diferentes sentidos. Em outras palavras, é preciso avaliar o comportamento espacial da variável regionalizada estudada. Esta avaliação é muito importante, pois através dela é possível encontrar a direção de maior continuidade, o que significa dizer que o mineral de interesse está melhor distribuído nesta ou naquela direção. Em caso de não haver descontinuidade, diz-se que o corpo mineralizado é isotrópico, isto é, ele possui a mesma variabilidade em todas as direções; esta é uma propriedade muito relevante, porque informa que o corpo mineralizado apresenta uma distribuição espacial bastante uniforme.

2.8.3 O Covariograma

O covariograma reflete a variabilidade média dos valores da amostra sobre seu valor médio comum, e sua equação é dada por:

$$cov(h) = \frac{1}{2n} \sum_{i=1}^{n} \{[(Z(x_i) - m)][Z(x_i + h) - m]\} \tag{2.2}$$

Onde n é o número de pares amostrais e m é a média de todos os valores amostrais.

Assim como o variograma, o diagrama de covariância ilustra o nível de continuidade e alcance dos valores amostrais, mas não possui um efeito de pepita. A faixa de influência dos valores das amostras ocorrem onde $cov(h)$ é zero (0), ou seja onde não há mais correlação espacial entre os dados amostrais. A desvantagem do covariograma é que ele requer uma estimativa do valor médio do depósito e assume que esse valor é o mesmo que o dos valores da amostra (FORKES, 1982), e isto gera uma instabilidade na análise variográfica, pois a média de uma localidade pode ser muito diferente de uma outra localidade da jazida mineral. Por esta razão ele não é muito aplicado para análise de estimativas de recursos minerais, salvo casos onde é muito difícil encontrar um variograma experimental que consiga definir o comportamento espacial do depósito mineral.

2.8.4 O Variograma Cruzado

Um variograma cruzado mede a continuidade espacial entre duas variáveis, A e B, para uma diatância variável h. A variância cruzada pode ser calculada mediante a equação a seguir:

$$\gamma_{cv}(h) = \frac{1}{2n} \sum_{i=1}^{n} \{[Z(x_{1i}) - Z(x_{1i} + h)][Z(x_{2i}) - Z(x_{2i} + h)]\} \tag{2.3}$$

Onde $Z(x_1)$ e $Z(x_2)$ são os valores da primeira e segunda variáveis respectivamente e n é o número de pares avaliados para cada incremento de h.

Portanto, o variograma, bem como suas respectivas variações, o covariograma e o variograma cruzado, representam uma indicação quanto à extensão ou não da variação local e espacial da variável regionalizada.

2.9 KRIGAGEM

A krigagem é um método de avaliação de recursos minerais, que faz uso da função variograma para calcular uma série de coeficientes de ponderação para valores amostrais que envolvem um bloco ou uma superfície mineralizada onde se deseja estimar o valor da variável naquele bloco ou superfície desconhecidas, (JOURNEL; HUIJBREGTS, 1978).

De maneira ilustrativa o procedimento para o cálculo do teor de minério em determinado ponto, pelo método da krigagem, é realizado pela interpolação dos pontos mais

próximos mediante o cálculo de uma matriz de covariância sobre a qual falaremos mais adiante. A figura 3 abaixo nos mostra esse procedimento.

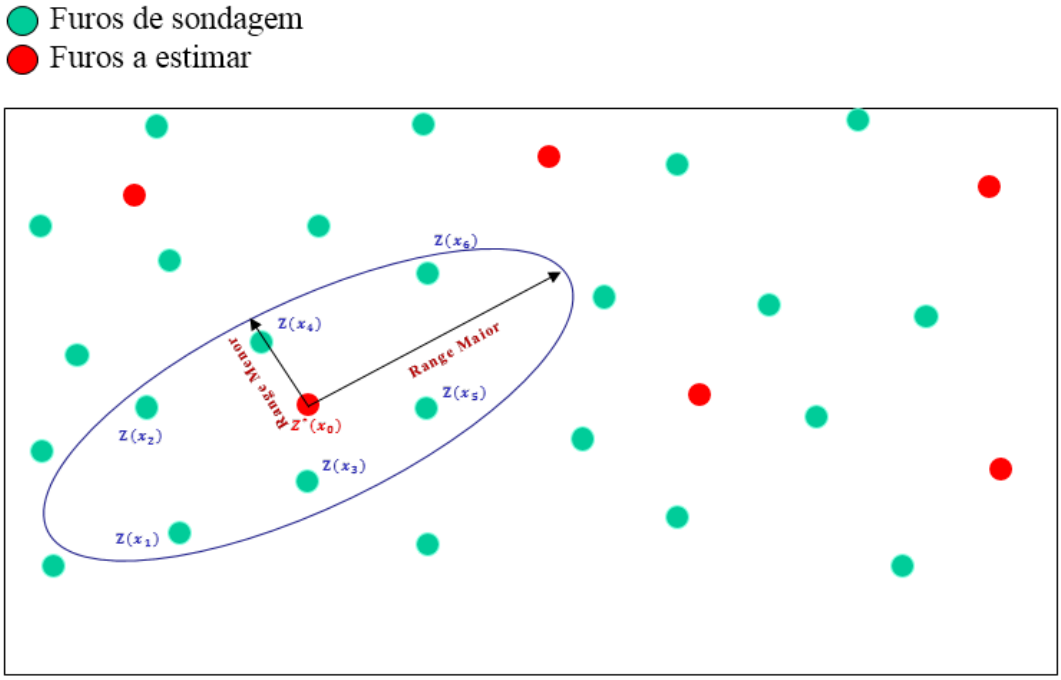

Figura 2 – Ilustração 2D de furos de sondagens

2.9.1 A Estimativa pelo Método da Krigagem

Consideremos o padrão de amostra na figura 3 em que se deseja estimar o valor no ponto *(vermelho)*, ou seja, é um valor desconhecido que precisa ser encontrado pelo método da krigagem. Cada uma das amostras terão influência entre si, como também irão influenciar o ponto desconhecido. O variograma teórico é o responssável por gerar os pesos referentes as amostras entre si e em relação ao ponto a ser estimado. O modelos teóricos mais conhecidos nas literaturas de geoestatística são representados pelas seguintes funções analíticas:

- Modelo Esférico

$$\gamma(h) = C_0 + C\left[1.5\frac{h}{a} - 0.5\left(\frac{h}{a}\right)^3\right] \qquad (2.4)$$
$$\gamma(h) = C_0 + C, \ \ \forall\ h \geq a$$

- Modelo Exponencial

$$\gamma(h) = C_0 + C\left[1 - exp\left(-\frac{h}{a}\right)\right] \qquad (2.5)$$

- Modelo Gaussiano

$$\gamma(h) = C_0 + C\left[1 - exp\left(-\left(\frac{h}{a}\right)^2\right)\right] \qquad (2.6)$$

- Modelo Cúbico

$$\gamma(h) = C_0 + C\left[7\left(\frac{h}{a}\right)^2 - \frac{35}{4}\left(\frac{h}{a}\right)^3 + \frac{7}{2}\left(\frac{h}{a}\right)^5 - \frac{3}{4}\left(\frac{h}{a}\right)^7\right] \qquad (2.7)$$
$$\gamma(h) = C_0 + C, \ \ \forall\ h \geq a$$

- Modelo Pentaesférico

$$\gamma(h) = C_0 + C\left[\frac{15}{8}\left(\frac{h}{a}\right) - \frac{5}{4}\left(\frac{h}{a}\right)^3 + \frac{3}{8}\left(\frac{h}{a}\right)^5\right] \qquad (2.8)$$
$$\gamma(h) = C_0 + C, \ \ \forall\ h \geq a$$

- Efeito Furo

$$\gamma(h) = C_0 + C\left[1 - \frac{sen\pi\left(\frac{h}{a}\right)}{\pi\left(\frac{h}{a}\right)}\right] \qquad (2.9)$$

A representação gráfica de cada um desses modelos é como a seguir:

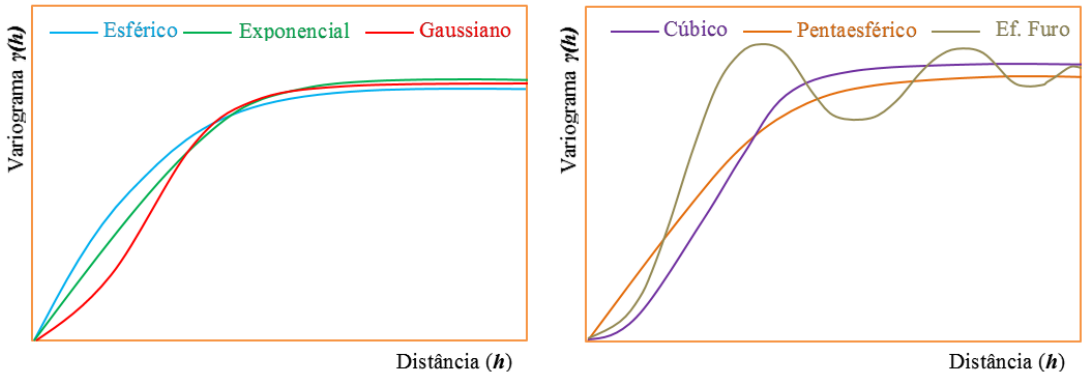

Figura 3 – Representação gráfica dos variogramas teóricos

Os variogramas teóricos supracitados representam a fundamentação teórica da krigagem, visto que essa metodologia de estimativa de recursos minerais estabelece esses modelos varigráficos analíticos para encontrar qualquer teor em pontos desconhecidos. Para encontrar esses teores esse método aplica uma matriz de covariância a qual tem como objetivo encontrar os ponderadores (*pesos*) que melhor predizem o valor desconhecido.

No entanto, para que possamos realizar uma estimativa por meio da aplicação do método da krigagem é necessário aplicar uma condição de contorno de não viés, ou seja, isto apenas pode ser alcançado se a média da estimativa krigada é a mesma que a média dos valores reais do depósito (DAVID, 2012), em outras palavras isto significa considerar a esperança matemática da variável regionalizada igual a média amostral do depósito, de forma que, no sistema matricial da krigagem sejam inseridos os multiplicadores de

Lagrange. Portanto, dado que a base para a estimativa por krigagem é o variograma, ou seja, a observação do comportamento da variância em função do incremento h, o método da krigagem consiste em encontrar os pesos entre as amostras bem como o ponto desconhecido, de maneira que a variância seja mínima. Portanto, método da krigagem pode ser definido como aquele que tem como objetivo minimizar a variância da estimativa, i.e. os coeficientes de ponderação são dispostos de modo a obter uma média ponderada que terá a menor estimativa do erro dada pela variância mínima.

2.9.2 Estimativa do Bloco de Minério

O exemplo anterior demonstra a avaliação de um valor pontual do corpo mineralizado. Em termos reais, isso é de uso limitado uma vez que os jazidas são avaliadas em termos de blocos de minério, cujo somatório dos possíveis blocos gerados dá origem ao modelo geológico de teores finalizado ou modelo de blocos de minério final. Logo, a krigagem deve ser capaz de estimar um bloco de minério, a fim de se obter um valor estimado para qualquer bloco discretizado no modelo geológico. Para efeito de cálculo é considerado que o teor representativo do bloco de minério encontra-se no centro dos respectivos blocos, o que vai gerar um cálculo matricial resultante em relação ao bloco de minério e não necessariamente a um ponto. Portanto, a variância da estimativa da krigagem para os valores dos blocos não deve considerar apenas a variância da estimativa do valor do bloco, mas também variância da estimativa das amostras reais dentro do bloco.

Com isto é possível produzir uma estimativa para cada bloco, como também atribuir um nível de confiança para a estimativa realizada.

2.9.3 O Efeito Viés

Como o objetivo da krigagem é produzir uma estimativa não enviesada e isto só pode ser alcançado se a média das estimativas krigadas é a mesma que a média dos valores reais. Portanto, um estimador perfeito produziria uma regressão linear dos valores reais contra valores estimados como sendo uma reta de inclinação '1', ou seja com um erro de estimativa de teores igual a zero, o que quer dizer que todos os pontos estariam sobre a curva de regressão. Mas, na prática, isso nunca pode ser alcançado devido a irregularidades nos dados e da impossibilidade de encontrar um estimador perfeito.

Sabendo-se que os blocos podem ser sub ou superdimencionado, em termos de seus teores reais medidos, isso geraria um efeito viés maior, e com a aplicação do método da krigagem esse efeito seria minimizado e o recurso mineral em estudo seria melhor aproveitado. Esta é a razão pela qual o método de krigagem é mais aplicado no campo da estimativa dos recursos minerais.

2.9.4 Modelagem Geológica

A modelagem geológica, segundo Rosemblueth & Weiner "Nenhuma parte substancial do Universo é tão simples que possa ser compreendida e controlada sem abstração. A abstração consiste em substituir a parte do universo em estudo por um modelo semelhante, porém de estrutura mais simples. Os modelos constituem, portanto, uma necessidade primordial de qualquer procedimento científico". Os modelos permitem a manipulação experimental de sistemas reais e possibilitam avaliar a maneira como a alteração de certos aspectos influencia o comportamento do mesmo. Um modelo deve reunir duas qualidades conflitantes: o realismo e a simplicidade, o realismo confere precisão à abstração do modelo real, enquanto a simplicidade permite sua manipulação experimental, (MARTINHO et al., 2006).

Desde a descoberta de uma jazida mineral até o momento de sua exaustão os modelos têm um papel fundamental na predição do comportamento do fluxo de minério bem como da geometria final da cava. A construção de um modelo geológico está fundamentada no conhecimento da resistência mecânica, dos indices físicos, interceptação de fraturas e mineralogia da jazida, onde estas informações são obtidas através de sondagem do corpo mineralizado. O modelo geológico de teores tem por finalidade calcular a distribuição de teor ao longo do corpo mineralizado e representa-los de forma qualitativa e quantitativa. Para a elaboração de um modelo geológico se faz um amplo uso da geoestatística que é entendida como um ramo da estatística que se concentra em conjuntos de dados distribuídos no espaço e/ou espaço-temporais. Em outras palavras é a ciência que trata os dados provenientes dos estudos geofísicos assim como de sondagem e os representa sob a forma de um modelo geológico, o qual é uma representação o tanto quanto possível mais próxima do real comportamento da jazida. Portanto a geo-modelagem é bastante usada para o reconhecimento e gerenciamento de recursos naturais, sendo as principais aplicações nos ramos da engenharia de minas e petróleo. Mas também é perfeitamente possível aplicar as técnicas de modelagem para resolver problemas de aquíferos, identificação de contaminantes em solo ou rios entre outros.

2.9.5 Qualidade da Modelagem

As técnicas mais usadas para se desenvolver modelos geológicos com maior acurácia até o momento são provenientes da geoestatística, mais especificamente dos métodos de krigagem (GOOVAERTS, 1999). O procedimento, uma vez tendo o banco de dados representativo daquilo que se quer avaliar, está fundamentado em, basicamente, quatro fases: análise comportamental dos dados, criação do modelo variográfico do corpo mineralizado estudado, estimativa por krigagem e validação dos resultados.

2.10 APRENDIZADO DE MÁQUINA

O campo da aprendizagem de máquina está preocupado com a questão de como construir programas de computador que melhoram automaticamente com a experiência. Nos últimos anos muitas aplicações bem sucedidas de aprendizado de máquina foram desenvolvidas, nos mais variados campos científicos o que nos permite inovar a cada dia de forma a atingir níveis mais elevados tanto em precisão como nas agilidade de obtenção dos resultados (BEKKERMAN; BILENKO; LANGFORD, 2011).

Portanto, a partir do que já foi mencionado percebemos que o principal da teoria do Aprendizado de Máquina consiste na programação de computadores de forma que possam ser capazes de aprender de forma semelhante ao humano. Se pudéssemos entender como programá-los para aprender, exatamente como um ser humano aprende, o impacto positivo na solução de problemas reais seria bastante significativo. Mas, percebemos que para alcançar o nível de interpretação equivalente a de um ser humano comum, a Inteligência Artificial precisa avançar um pouco mais. Contudo, vemos que, mesmo diante das limitações interpretativas dos algoritmos computacionais, vemos que o avanço se dá com a possibilidade de melhorar automaticamente com a experiência, ou seja, com a capacidade que a máquina tem de aprender com os dados que nela são inseridos para o treinamento experimental. Imaginemos, por exemplo, computadores aprendendo com registros para diagnosticar procedimentos médicos mais eficazes no tratamento de novas novas doenças, ou casos que aprendem com a experiência para otimizar os custos de energia com base em padrões particulares de uso de seus ocupantes, ou assistentes pessoais de software que avaliam a evolução dos interesses de seus usuários etc. Uma compreensão bem-sucedida de como fazer computadores aprenderem abriria muitos novos usos de computadores e novos níveis de competência e personalização.

O desafio do Aprendizado de Máquina consiste na criação de algoritmos que sejam eficazes para certos tipos de tarefas de aprendizagem, pois sabemos que o comportamento de um determinado fenômeno pode ser melhor explicado por uma determinada classe de algoritmos, uma vez que façamos diversos experimentos que nos mostrem a eficácia nos resultados obtidos, de forma que, a partir desses resultados saibamos a classe de algoritmos que melhor representa o fenômeno de interesse, para isto também é necessário uma compreensão teórica da aprendizagem, pois esta compreensão fornecerá a base para a justificativa da aplicação bem sucedida. No campo conhecido como mineração de dados, por exemplo, os algoritmos de aprendizado de máquina estão sendo usados rotineiramente para descobrir informações valiosas de grandes bancos de dados comerciais contendo registros de manutenção de equipamentos, pedidos de empréstimo, transações financeiras, registros e afins. Como nossa compreensão de computadores continua amadurecendo, parece inevitável que a aprendizagem automática desempenhe um papel cada vez mais dinâmico e automatizado. A partir dessas experiências já bem implementadas podemos migrar ou adatar a outros campos de estudos de forma que através de algumas

modificações nas implementações dos códigos computacionais possamos atingir resultados satisfatórios na análise de dados de uma natureza completamente diferente, que neste caso é naturaza geológica, mas que possue um comportamento semântico semelhante aos já avaliados tradicionalmente no Aprendizado de Máquina.

Algumas realizações específicas fornecem um vislumbre do estado da arte: programas que foram desenvolvidos para o reconhecimento com sucesso de palavras faladas (FAHLMAN; LEBIERE, 1990), previsão de taxas de recuperação em pacientes com pneumonia (COOPER et al., 1997), detectar o uso fraudulento de cartões de crédito, dirigir veículos autônomos em rodovias públicas (BOJARSKI et al., 2016) e jogar jogos como gamão ou xadrex em níveis que se aproximam ou até superam o desempenho dos campeões mundiais humanos (TESAURO, 1995). Foram desenvolvidos resultados teóricos que caracterizam os fundamentos e a relação entre o número de exemplos de treinamento observados, o número de hipóteses em consideração e o erro esperado nas hipóteses aprendidas. Portanto, a leitura dessas aplicações foram de grande importância para o encorajamento para construir um método de AM que fosse eficiente e eficaz para a análise de recursos minerais. As aplicações dos algoritmos em problemas reais, bem como a sua teoria no estudos de sistemas computacionais aumentou a taxa de progresso significativamente na última década. Este trabalho apresenta um campo do aprendizado de máquina, que é o campo do aprendizado supervisionado, o qual necessita de um "professor", ou seja, precisa possuir a variável independente como objeto de estimativa ou predição, para que o algoritmo possa reconhecer o padrão que se deseja modelar, e consequentemente fornecer o resultado esperado, que em nosso caso, é um modelo geológico da jazida mineral.

3 A SOLUÇÃO PROPOSTA

3.1 JUSTIFICATIVA PARA A APLICAÇÃO DE APRENDIZADO DE MÁQUINA

Apesar dos estimadores por krigagem possuírem uma ampla gama de aplicação em diversos segmentos, sua capacidade estimativa depende grandemente da qualidade dos dados utilizados. Os dados utilizáveis aplicam-se à presença de dados bons e suficientes para mapear a estrutura de correlação espacial. O seu desempenho também é sensivelmente melhor quando existe uma relação linear entre os padrões de entrada e saída. Na vida real, no entanto, isso é extremamente improvável. Apesar de existir uma série de versões de krigagem, como krigagem log-normal e krigagem de indicadores, que aplicam certas transformações específicas para capturar relacionamentos não lineares, podem não ser eficientes o suficiente para capturar a natureza ampla da não linearidade espacial, (DUTTA et al., 2010). Além da dificuldade de se trabalhar com dados não lineares, a geoestatística possui outra limitação, que é aquela na qual a jazida não possui estrutura, ou seja, ocorre uma variabilidade infinita dos dados, o que quer dizer que o variograma não possui patamar ou não se estabiliza em um valor. Portanto, o uso de Redes Neurais Artificiais representa um caminho que pode ser trilhado a fim de obter uma melhor performance para as estimativas de recursos minerais. A aplicação dos algoritmos não lineares e em particular máquina de vetores de suporte - Support Vector Machine (SVM) para a estimativa de recursos minerais foi realizada como um estudo inicial para a avaliar o comportamento dos resultados obtidos com o procedimento de aprendizado de máquina comparado ao método tradicional de krigagem. O objeto de estudo para esta análise inicial foi uma jazida de minério de ferro, mas vale lembrar que esta metodologia de aprendizado de máquina tem a finalidade de ser aplicada a qualquer mineral, bem como a bancos de dados que possuam mais de um minério de interesse.

3.2 KRIGAGEM COMO UM PROCESSO GAUSSIANO

O procedimento de krigagem é baseado em um método de interpolação, no qual os valores são modelados por um processo gaussiano que segue a covariância anterior. Este procedimento também é conhecido como predição de Wiener - Kolmogorov, uma vez que este princípio é para minimizar o Erro do Estimador Quadrático Médio, o que implica o conhecimento explícito da função de covariância (BROVELLI; SANSO; VENUTI, 2003). A estatística espacial trata a covariância como uma função que descreve a medida de dependência espacial de um processo estocástico, e uma vez que esta função representa um processo estacionário, isto é, tem um $Sill$, pode ser relacionado com o variograma geral,

através da equação:

$$\gamma(h) = \frac{1}{2V} \iiint_V [f(M+h) - f(M)]^2 \, dV \tag{3.1}$$

Onde M é um ponto no campo geométrico, isto é, M representa as coordenadas de latitude, longitude e profundidade, $f(M)$ é o valor deste ponto e h é a distância entre cada amostra observada (MATHERON, 1963). Assim, este trabalho foi feito usando a abordagem acima mencionada, associada a , isto é, a média da krigagem é calculada usando o vizinho mais próximo n coberto pelo *range* do variograma. A estimativa do recurso mineral é, através do processo geoestatístico, como segue:

$$Z^*_{ko}(\mathbf{x}_0) = \sum_{i=1}^{n} \gamma_i Z(\mathbf{x}_i) \tag{3.2}$$

Portanto, dado $Z(\mathbf{x}_i)$ que representa o teor da coordenada $\mathbf{x}_i$, é possível calcular a nota do $\mathbf{x}_0$ que é a nota de coordenada desconhecida; por meio da equação acima, onde γ_i é calculado pela seguinte equação:

$$\begin{bmatrix} \gamma(\mathbf{x}_1 - \mathbf{x}_1) & \gamma(\mathbf{x}_1 - \mathbf{x}_2) & \cdots & \gamma(\mathbf{x}_1 - \mathbf{x}_n) & 1 \\ \gamma(\mathbf{x}_2 - \mathbf{x}_1) & \gamma(\mathbf{x}_2 - \mathbf{x}_2) & \cdots & \gamma(\mathbf{x}_2 - \mathbf{x}_n) & 1 \\ \vdots & \vdots & & \vdots & \vdots \\ \gamma(\mathbf{x}_n - \mathbf{x}_1) & \gamma(\mathbf{x}_n - \mathbf{x}_2) & \cdots & \gamma(\mathbf{x}_n - \mathbf{x}_n) & 1 \\ 1 & 1 & \cdots & 1 & 0 \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_n \\ -\mu \end{bmatrix} = \begin{bmatrix} \gamma(\mathbf{x}_0 - \mathbf{x}_1) \\ \gamma(\mathbf{x}_0 - \mathbf{x}_2) \\ \vdots \\ \gamma(\mathbf{x}_0 - \mathbf{x}_n) \\ 1 \end{bmatrix}$$

O Processo Gaussiano é um procedimento estocástico, no qual as variáveis aleatórias são indexadas por tempo ou espaço, de modo que toda coleção finita dessas variáveis aleatórias é governada por uma distribuição normal, ou seja, toda combinação linear finita de variáveis aleatórias é normalmente distribuída. O Processo Gaussiano é considerado como um procedimento de aprendizado de máquina que usa o método de aprendizagem Aprendizado Lento; onde a medida de similaridade entre os pontos, ou seja, as variáveis aleatórias, utilizadas para prever pontos não vistos são obtidas por meio de um *Kernel Function*, que neste caso é caracterizado por três funções principais, como segue:

$$\text{Modelo Exponencial} \quad \gamma(h) = C_0 + C\left[1 - exp\left(-\frac{h}{a}\right)\right] \tag{3.4}$$

$$\text{Moldelo Esférico} \quad \gamma(h) = C_0 + C\left[1.5\frac{h}{a} - 0.5\left(\frac{h}{a}\right)^3\right] \tag{3.5}$$

$$\text{Modelo Gaussiano} \quad \gamma(h) = C_0 + C\left[1 - exp\left(-\left(\frac{h}{a}\right)^2\right)\right] \tag{3.6}$$

Esses modelos devem cobrir aproximadamente 80% do comportamento do minério no subsolo (GOOVAERTS, 2000). Vê-se que, embora esses modelos possam representar o que está oculto no subsolo, eles têm algumas limitações, uma vez que a natureza pode representar algumas anomalias que os modelos citados não conseguem entender.

3.3 UMA ABORDAGEM DO APRENDIZADO DE MÁQUINA

Todo o processo de inferência espacial começa com a coleta de uma amostra composta de n pontos de dados que são vetores em R^3 juntamente com os respectivos valores dos conteúdos. Estes precisam ser de fato representativos do fenômeno espacial em estudo em termos de distribuição espacial e variabilidade, porque a qualidade da estimativa está diretamente ligada à qualidade dos dados que são usados para representar o corpo mineralizado. Portanto, nesta análise, as bases utilizadas foram Ferro, Ouro e Níquel, onde o método da krigagem foi comparado com os métodos de Aprendizado de Máquina.

Os algoritmos de aprendizado de máquina são modelos adaptativos que são capazes de ajustar seus parâmetros quando os dados de treinamento estão disponíveis para prever rótulos de pontos de dados não vistos (BISHOP, 2006). Tais algoritmos foram empregados com sucesso em uma ampla gama de problemas de previsão (MENDES-MOREIRA et al., 2012; WU et al., 2008), tais como: Aplicações ecológicas (CRISCI; GHATTAS; PERERA, 2012), genética (LIBBRECHT; NOBLE, 2015), robótica (RANI et al., 2006), séries temporais financeiras (KIM, 2003), pontuação de crédito (WANG et al., 2011).

Seja um conjunto de pontos de treinamento (instâncias) $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\} \subset \mathcal{X} \times \mathbb{R}$, onde $\mathbf{x}_i = (x_1, \ldots, x_d)^T$ denota um vetor de atributos de operação de mineração d. Em nosso estudo, o espaço de entrada $\mathcal{X} = \mathbb{R}^d$. Os rótulos y_i representam a função de destino a ser estimada. Um regressor de aprendizado de máquina aprende com $\mathcal{D}$ para generalizar o rótulo y de um ponto não visto $\mathbf{x}$. O treinamento de tal regressor consiste em encontrar uma função $f : \mathbb{R}^d \to \mathbb{R}$ no espaço da hipótese possível $\mathcal{H}$ que resolve

$$f^*(\mathbf{x}) = \text{argmin}_{f \in \mathcal{H}} \mathcal{L}(\mathbf{X}), \tag{3.7}$$

onde $\mathcal{L}(f, y)$ é uma função de perda que mede a similaridade entre o rótulo previsto e o rótulo de destino. A abordagem proposta consiste em algumas etapas que são mostradas pela figura 4:

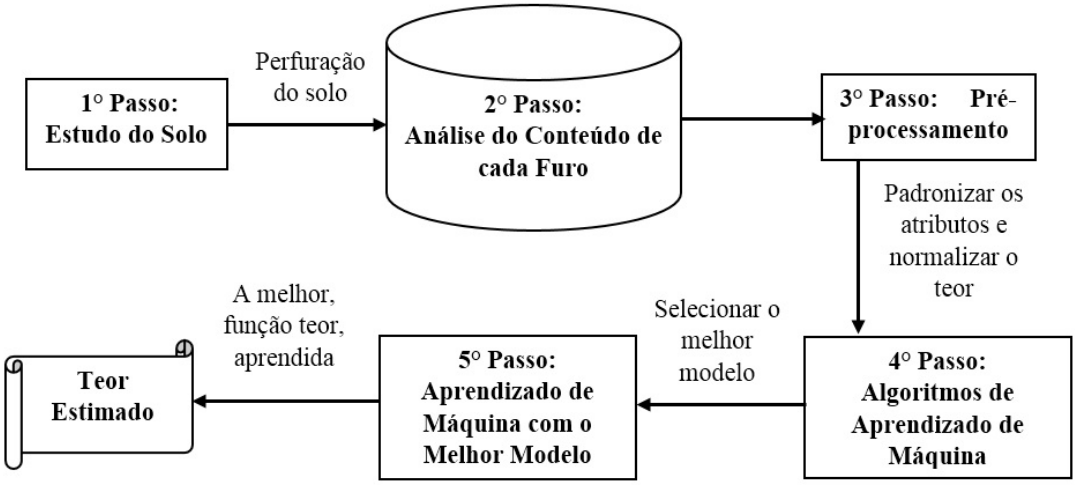

Figura 4 – Abordagem de Aprendizado de Máquina para estimar o teor de recurso mineral

Seguindo a metodologia citada, foram aplicados dez modelos de Aprendizado de Máquina, para que sejam comparados com a metodologia tradicional.

3.4 REGRESSORES UTILIZADOS

Os modelos multivariados avaliados em diversos campos de pesquisa envolvem análise do relacionamento entre múltiplas variáveis explicativas e, em alguns casos, múltiplas variáveis dependentes. Grande parte das pesquisas delineadas para examinar o efeito exercido por duas ou mais variáveis independentes sobre uma variável dependente utiliza a análise de Regressão Múltipla (RM). Essa regressão como sendo um conjunto de técnicas estatísticas que possibilitam a avaliação do relacionamento de uma variável dependente com diversas variáveis independentes (ABBAD; TORRES, 2002). O resultado final de uma RM, seja ela linear ou não, é uma equação que apresenta a melhor estimativa de uma variável dependente a partir de diversas variáveis independentes. Este tipo de equação representa um modelo aditivo, no qual as variáveis preditoras somam-se à explicação da variável dependente. Portanto, dentro desse contexto, a nossa análise, para a realização desse trabalho, está fundamentada no estudo e implementação de dez métodos de AM, os quais serviram para avaliar o real comportamento das jazidas minerais avaliadas nesse estudo, e a partir dos resultados selecionar a categoria de algoritmos que melhor representa a distribuição espacial dos recursos minerais de interesse. Para cada um dos métodos avaliados seram pontuados os diferentes aspectos teóricos que governam suas respectivas particularidades.

Neste trabalho foram avaliados três classes de algoritmos de AM, os de caráter linear, não-linear e os ensenbles; sobre os quais falaremos as suas principais particularidades, as quais serviram da base para a geração da solução do problema analisado nessa tese.

3.5 MÉTODOS LINEARES

A seguir serão apresentados os métodos lineares que foram utilizados na busca da solução do problema proposto.

3.5.1 Regressão Linear

A Regressão Linear (RL) em sua forma mais simples relaciona uma variável independente X com a sua respectiva variável dependente Y, por meio de uma reta que melhor se ajusta aos dados avaliados. Em nossa abordagem o objetivo é, exatamente, o mesmo do que já foi mencionado, apenas com uma diferença, pois agora temos um vetor de três coordenadas, as quais representam as variáveis independentes, e temos como a quarta dimensão a variável que queremos estimar que em nosso caso é o teor no respectivo minério. Portanto, estamos agora diante de um problema multivariado o qual está relacionando três componentes para avaliar um resultado, que neste caso é a avaliação do comportamento espacial do mineral de interesse no subsolo. A representação analítica do modelo de regressão matemática multipla avaliado neste trabalho é mostrada a seguir:

O modelo mais simples para estimar o conteúdo de minério no subsolo consiste na avaliação de uma combinação linear a qual pode correlacionar as variáveis do problema através de uma função cujas varáveis independentes possuem grau um, o que implica uma regressão linear. Seja o conjunto de pontos $\{y_i, x_{i1}, \cdots, x_{ip}\}_{i=1}^{n}$ onde os x_ij e y_i representam as variáveis independentes e dependentes respectivamente e cuja relação entre as variáveis dependente e independentes é estritamente linear. Portanto, a equação que representa este fenômeno linear é: $y_i = \beta_0 1 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i$. Sendo o erro residual de uma amostra dada por $e_i = y_i - \hat{y}_i$, o nosso objetivo é encontrar os respectivos coeficientes $\{\beta_i, i = 1, ...n\}$ de forma que o erro residual seja minimizado. Logo, o trabalho realizado pelo AM consiste na obtenção dos melhores coeficientes, ou seja, aqueles que irão gerar o erro mínimo, e isto é feito mediante a solução da seguinte equação:

$$\left(\hat{\alpha}, \hat{\beta}\right) = \arg min \left\{ \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 \right\} \tag{3.8}$$

Em sua forma vetorial vem:

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{X}^t \boldsymbol{X})^{-1} \boldsymbol{X}^t \boldsymbol{y} \tag{3.9}$$

Onde $\boldsymbol{X}$ é o vetor de variáveis independentes e $\boldsymbol{y}$ representa o vetor da variável dependente, t e -1 representam o símbolo de matriz transposta e inversa respectivamete Esse modelo foi aplicado para ver a natureza do conjunto de dados, isto é, se demonstra algum tipo de linearidade, de forma que fosse possível realizar um inferência linear, no que concerne ao conjunto de dados, bem como validar o seu comportamento no subsolo.

3.5.2 Regressão *Lasso*

Este método de regressão foi aplicado para verificar o nível de relacionamento das variáveis independentes, uma vez que este método utiliza a penalidade l^1 tanto para ajuste quanto para penalização dos coeficientes. Este é um método de regularização dos coeficientes da variáveis independentes, isto é feito a fim de avaliar as variáveis que são mais representativas para explicar o problema em análise. Portanto, a regressão Lasso é o que chamamos de método de regressão penalizado, usado frequentemente em aprendizagem de máquina para selecionar um subconjunto de variáveis. É um método supervisionado de aprendizagem de máquina. Especificamente, LASSO é um método de *Shrinkage* e Seleção de Variáveis para modelos de regressão linear. LASSO é o acrônimo para *"Least Absolute Selection and Shrinkage Operator"* (mínimo encolhimento absoluto e operador de seleção, em tradução livre)

Esta metodologia é aplicada mais especificamente para avaliar o grau de explicabilidade das variáveis independentes com relação a variável dependente, em outras palavras, podemos dizer que esse método impõe uma restrição na soma dos valores absolutos dos parâmetros do modelo, cuja soma possui uma constante especificada como um limite superior. Esta restrição faz com que os coeficientes de regressão para algumas variáveis se encolham e sempre que possível sejam direcionadas para a nulidade ou bem próximo dela. Este representa o processo de encolhimento onde se configura a essência desse método. Com este processo de encolhimento é possível auferir uma melhor interpretação do modelo e identificar as variáveis mais fortemente associadas com a variável alvo ou dependente correspondente. Esse é o processo de seleção das variáveis e mediante esse processo é possível obter o subconjunto de preditores ou variáveis explicativas ou independentes que minimiza o erro de predição.

Logo, para a apresentação das equações que representam esse método, consideremos o conjunto de pontos $\{y_i, x_{i1}, \cdots, x_{ip}\}_{i=1}^{n}$ onde a relação entre as variáveis dependentes e independentes é linear, e é dada pela equação: $y_i = \beta_0 1 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i$. O erro residual, que é o objeto de minimização, é dado por: $e_i = y_i - \hat{y}_i$, esta difere da regressão linear apenas pela restrição limitante, que é mostrada juntamente com sua função objetivo, como segue:

$$\left(\hat{\alpha}, \hat{\beta}\right) = \text{arg}min \left\{ \sum_{i=1}^{n} \left(y_i - \alpha - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 \right\} \tag{3.10}$$

$$\text{sujeito a} \sum_j |\beta_j| \leq t$$

Sua forma vetorial é:

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{X}^t \boldsymbol{X} + \lambda \boldsymbol{W})^{-1} \boldsymbol{X}^t \boldsymbol{y} \tag{3.11}$$

Onde $\boldsymbol{W}$ é uma matriz diagonal da norma dos elementos β_j e λ são pesos calculados de forma que $\sum |\beta_j|^* = t$.

3.5.3 Regressão *ElasticNet*

Dados do mundo real aplicados em estudos de simulação mostram que o *ElasticNet* (EN) é bastante eficiente para a criação de modelos de regressão multipla, onde o objetivo é dar uma maior precisão ao modelo final, e este método, frequentemente, supera o Lasso, enquanto desfruta de uma excelente capacidade de esparsidade, ou seja, na seleção de das variáveis que mais conseguem explicar o modelo finalizado, (ZOU; HASTIE, 2005). Este método de regressão é muito semelhante ao Lasso. Ele difere, basicamente, porque trabalha simultaneamente com a seleção automática de variáveis e o contínuo encolhimento, isto é, podemos selecionar grupos de variáveis correlacionadas de maneira a melhorar o resultado final do modelo, deixando-o com o mínimo de variáveis independentes. Para a aplicação desse método considera-se o mesmo conjunto de entrada supracitado anteriormente, em RL e Lasso. Sendo a função objetivo a mesma de RL e Lasso, mas diferindo apenas pela restrição, temos que a formulação matemática do EN é dada como mostrado a seguir:

$$\left(\hat{\alpha}, \hat{\beta}\right) = \text{arg}min \left\{ \sum_{i=1}^{n} \left(y_i - \alpha - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 \right\} \tag{3.12}$$

$$\text{sujeito a} (1-\alpha) \sum_{j=1}^{p} |\beta_j| + \alpha \sum_{j=1}^{p} \beta_j^2 \leq t$$

3.6 MÉTODOS NÃO LINEARES

A aplicação de métodos não lineares consiste na hipótese de que o comportamento da distribuição espacial dos dados em subsolo pode ser modelada de forma não linear, ou seja, dada a possibilidade de não linearidade da distribuição dos dados é plausível que apliquemos métodos de AM não lineares. Os métodos avaliados para busca da solução do problema analisado são mostrados a seguir, juntamente com a base teórica que fundamenta cada um dos métodos avaliados neste trabalho.

3.6.1 Árvore de Regressão - *CART*

A utilização de árvores de decisão binárias para classificação pode ser considerada uma abordagem não-paramétrica para reconhecimento de padrões, que em nosso caso específico o padrão representa um vetor de atributo o qual tem como variável alvo o teor de minério. A linha da matriz que contém os atributos X, Y e Z juntamente com o teor recebe o nome de instâncias. Utilizando-se uma escala de cores podemos avaliar os padrões da variável de interesse e interpretá-los como uma escala de cores na qual estão alocados os grupos de teores do mineral de interesse. Uma árvore de decisão faz uma representação hierárquica do espaço de feições, onde padrões são alocados aos conjuntos ou classes w_j conforme o resultado encontrado depois de percorridos os ramos da árvore. Árvores de decisão binárias consistem de repetidas divisões do espaço de feições em dois subgrupos descendentes que

terminam em *nós* associados às classes w_j. Na terminologia de árvores de decisão os subgrupos do espaço de feições são definidos através de *nós*. Uma árvore de decisão com alto poder preditivo e um pequeno número de nodos constitui uma situação altamente desejável. O tipo de árvore de decisão tratada nesse trabalho foi o $Classification\ And\ Regression\ Trees$ - $CART$. A escolha do algoritmo CART resultou da sua possibilidade de melhor generalização dos dados, visto que a partição dos *nós* se dá de forma binária. Isto é muito bom do ponto de vista do aprendizado de máquina, uma vez que o objetivo final é encontrar um modelo que seja capaz de prever dados alheios aos bancos de dados utilizados para o treinamento do método.

3.6.2 Funcionamento da Técnica CART

A técnica CART constrói uma árvore de decisão binária a partir de uma amostra de treinamento formando partições na amostra. Esse fato faz com que os tamanhos amostrais dos subgrupos formados por tais partições decresçam, exigindo um tamanho amostral razoável para obtenção de bons resultados, (RIPLEY, 1994). A árvore de decisão inicia com o *nó* raiz t contemplando aquela variável do espaço de feições que minimiza o grau de impureza de dois nodos irmãos. A medida do grau de impureza do *nó* t denotada por $i(t)$ é determinada pela seguinte expressão:

$$i(t) = -\sum_{j=i}^{k} p(w_j \mid t) log[p(w_i \mid t)] \tag{3.13}$$

Onde $p(w_j \mid t)$ é a proporção de padrões alocados ao conjunto ou classe w_j no *nó* t. Todos *nós* não terminais dividem-se em outros dois *nós*, que podemos chamar de t_l e t_r onde l e r são, respectivamente, os lados esquerdo e direito, da partição binária. Sendo as partes correspondentes às proporções de observação dos *nós* esquerdo e direito, denotadas por p_l e p_r respectivamente. Podemos realizar a melhor partição s de crescimento da árvore mediante a diferença máxima da seguinte equação:

$$\Delta i(s,t) = i(t) - [p_l i(t_l) + p_r i(t_r)] \tag{3.14}$$

Portanto, o crescimento da árvore de decisão se dá por sucessivas divisões até que não haja mais possibilidade de decréscimo significante no grau de impureza por meio de uma divisão s. Quando isto ocorre, o *nó* t não é mais dividido, sendo automaticamente transformado em um *nó* terminal. A classe w_j associada ao *nó* terminal t é aquela que maximiza a probabilidade condicional $p(w_j \mid t)$.

3.6.3 Máquinas de Vetores de Suporte

Uma máquina de vetores de suporte ou $Support\ Vector\ Machine$ - SVM, de uma forma simplificada, é um método de aprendizagem de máquina que dado um vetor de entrada,

o algoritmo tenta classificá-lo em uma das categorias ou classes existentes em nosso problema. Para que uma máquina de vetores de suporte seja eficaz, primeiramente é necessário utilizar um conjunto de dados de entrada e de saída de treinamento para construir o modelo de máquina de vetores de suporte que pode ser utilizado para predição de novos dados.

Uma máquina de vetores de suporte desenvolve esse modelo tomando as entradas de treinamento, mapeando elas no espaço multidimensional e utilizando regressão para encontrar um hiperplano ótimo, ou seja, esse hiperplano é uma superfície em espaço de n dimensões que separa em classes o conjunto de dados de entrada. Uma vez que a máquina de vetores de suporte tenha sido treinada, ela é capaz de avaliar quaisquer novas entradas em relação ao hiperplano divisor e classificá-las entre as classes existentes.

De uma forma intuitiva pode-se dizer que uma máquina de vetores de suporte é essencialmente uma máquina de entrada/saída, ou seja, dada uma entrada e, com base no modelo desenvolvido através do treinamento, ela devolverá uma saída, que corresponde ao respectivo vetor de atributos ou vetor de entrada. O número de entradas para qualquer máquina de vetores de suporte especificada, teoricamente, varia de um a infinito enumerável. Entretanto, em termos práticos, a capacidade computacional limita a quantidade de entradas que pode ser utilizada, (ALEEM; CAPRETZ; AHMED, 2015).

Para a solução desse problema foi aplicado o ϵ-Support Vector Regression (ϵ-SVR) de modo a aproximar a função que governa o teor de minério no subsolo da região avaliada. Este método objetiva aprender uma função do tipo: $f(\mathbf{x})$, a qual desvia ao máximo ϵ da função objetivo, para todos os dados de treinamento, e isto é feito da forma mais plana possível, ou seja, não é permitido erros de ordem maior que ϵ. Esta formulação é importante se nós queremos limitar o teor de minério através do ϵ na predição do recurso mineral.

SVR são, essencialmente algoritmos de treinamento lineares, baseados no conceito de máxima margem, o que o torna um algoritmo mais robusto para o uso de cálculo não lineares é a sua função Kernel, dada por $\phi(\mathbf{x}_i)$, cuja formulação principal está representada pela equação a seguir:

$$f(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + b, \tag{3.15}$$

onde $\phi(\mathbf{x}_i)$ representa a transformação ou mapeamento $\phi : \mathcal{X} \rightarrow \mathcal{F}$ do espaço de entrada $\mathcal{X}$ em um espaço de característica $\mathcal{F}$. Tal transformação permite o linear ϵ-SVR ser empregado para o comportamento de dados não lineares.

O nivelamento de $f(\mathbf{x})$ é governado pela norma $||\mathbf{w}||^2 = \mathbf{w}^T\mathbf{w}$. A otimização de $f(\mathbf{x})$ é um problema convexo como a seguir:

$$\begin{aligned} &\text{minimize} && \tfrac{1}{2}||w||^2 \\ &\text{subject to} && \begin{cases} y_i - \mathbf{w}^T\phi(\mathbf{x}_i) \leq \epsilon, & i = 1, \ldots, n \\ \mathbf{w}_i^T\phi(\mathbf{x}) + b - y_i \leq \epsilon, & i = 1, \ldots, n \end{cases} \end{aligned} \tag{3.16}$$

Tal procedimento assume que $f(\mathbf{x})$ existe para todas as instâncias x_i com ϵ de precisão. Frequentemente, essa restrição não é rigorosamente atendida, visto que são permitidos erros a fim de se obter uma solução, (BENNETT; MANGASARIAN, 1992), (CORTES; VAPNIK, 1995). Tais erros são representados pelas variáveis de folga ξ, ξ^*, as quais são introduzidas para lidar com o potencial não factível das restrições. Portanto, a equação 3.17 pode ser reescrita como:

$$\begin{aligned} &\text{minimize} && \tfrac{1}{2}||w||^2 + C\sum_i^n (\xi_i + \xi_i^*) \\ &\text{subject to} && \begin{cases} y - \mathbf{w}^T\phi(\mathbf{x}) \leq \epsilon + \xi_i, & i = 1, \ldots, n \\ \mathbf{w}^T\phi(\mathbf{x}) + b - y \leq \epsilon + \xi_i, & i = 1, \ldots, n \\ \xi_i, \xi_i^* \geq 0, \end{cases} \end{aligned} \tag{3.17}$$

Onde a restrição de caixa $C > 0$ denota o balanço entre o nivelamento de f e os erros maiores do que ϵ que são permitidos. Tal restrição ajuda a previnir o overfitting.

3.6.4 k Vizinhos mais Próximos - $k - NN$

O método k-Nearest Neighbor - KNN foi aplicado a fim de descobrir alguma semelhança com o método clássico, uma vez que este estima o rótulo ou variável alvo que em nosso caso é representado pelo teor do minério de interesse, por meio de seus vizinhos mais próximos. Este é um método não-paramétrico, que é usado tanto para classificação quanto para regressão. Em ambos os casos, as entradas consistem nos exemplos de treinamento k mais próximos do conjunto das variáveis alvo. A estimativa que foi realizada com o uso desse método é dada pelo pseudocódigo mostrado seguir:

Algoritmo 1: k - Vizinho mais Próximos

1: Prepare o conjunto de entrada e saída.
2: Informe o valor de k.
3: Calcular a distância euclidiana para todas as amostras.
4: Determinar o conjunto dos k vizinhos cujas distâncias sejam mais próximas da variável alvo.
5: O rótulo com o maior número de representantes, no conjunto de vizinhos mais próximos, será o escolhido.

3.7 APRENDIZADO DE COMITÊ

As técnicas de comitê são técnicas que criam múltiplos modelos e depois os combinam para produzir resultados melhores (SEWELL, 2008). No aprendizado supervisionado, a simplicidade computacional é obtida distribuindo a tarefa de aprendizado entre vários especialistas, que, por sua vez, dividem o conjunto de entrada em um conjunto de subespaços. Essa combinação forma o que é chamado de aprendizado de comitê ou método ensemble. Essa metodologia funde o conhecimento adquirido por especialistas para chegar a uma decisão global que, na maioria das vezes, é superior àquela alcançável por qualquer um deles agindo sozinho. Segue uma ilustração dada pela figura 5.

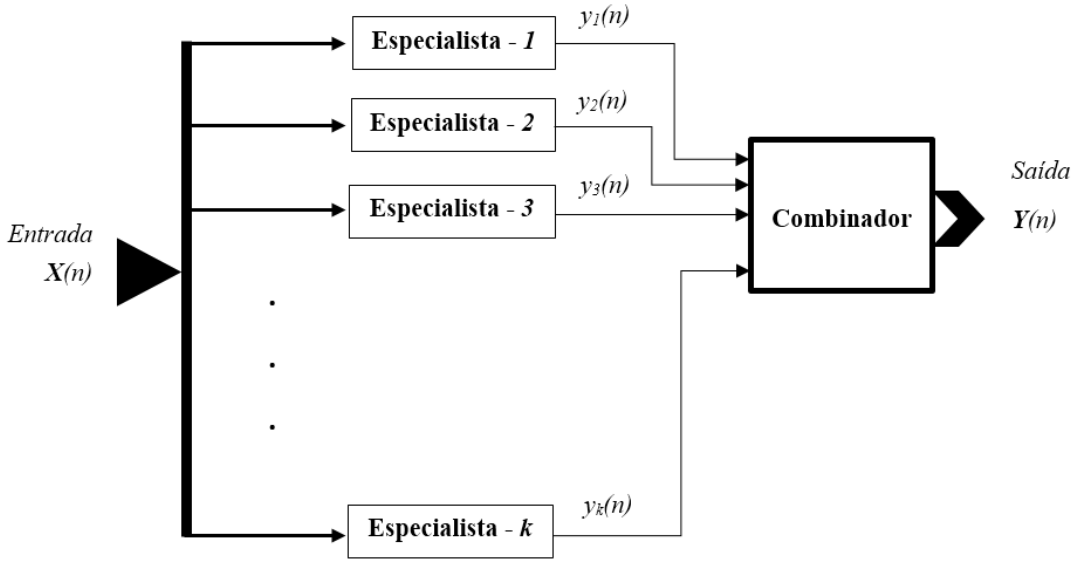

Figura 5 – Diagrama de bloco de uma máquina de comitê

3.8 MÉTODOS ENSEMBLE

Os métodos ensemble são algoritmos de aprendizado de máquina que constroem um conjunto de classificadores ou regressores, que é o nosso caso, e depois classifica um novo vetor de entrada tomando o voto ponderado das predições. A condição necessária e suficiente para um conjunto de classificadores ou regressores ser mais preciso do que qualquer um dos seus membros individuais é o fato de que o conjunto de classificadores é representado por uma combinação desses classificadores, o que gera um resultado mais preciso (HANSEN; SALAMON, 1990).

Um algoritmo de aprendizado de máquina pode ser visto como um ente matemático que tenta buscar um espaço H de hipóteses para identificar a melhor hipótese no espaço de solução do problema. O problema estatístico surge quando a quantidade de dados de treinamento disponível é pequena, comparada a dimensão do espaço de hipóteses. Em não existindo a abundância de dados disponíveis, o algoritmo de aprendizado de máquina

pode encontrar muitas hipóteses diferentes em H que gerarão diversos resultados sobre o conjunto de treinamento. Ao construir um conjunto de classificadores ou regressores provenientes das diversas hipóteses geradas o algoritmo ensemble pondera esses classificadores de forma a reduzir o risco de escolher um classificador errado. A figura 6 descreve essa situação. A curva externa representa o espaço de hipóteses H, a curva interna representa o conjunto de hipóteses que gerará uma boa acurácia nos dados de treinamento, a função f é a nossa hipótese verdadeira, e através da ilustração vemos que ao ponderar os classificadores h_1, h_2 e h_3 podemos obter uma boa aproximação para o valor real f. Portanto, do ponto de vista estatístico a aplicação do método ensemble representa uma maior vantagem se comparado ao tratamento dos dados mediante apenas um regressor.

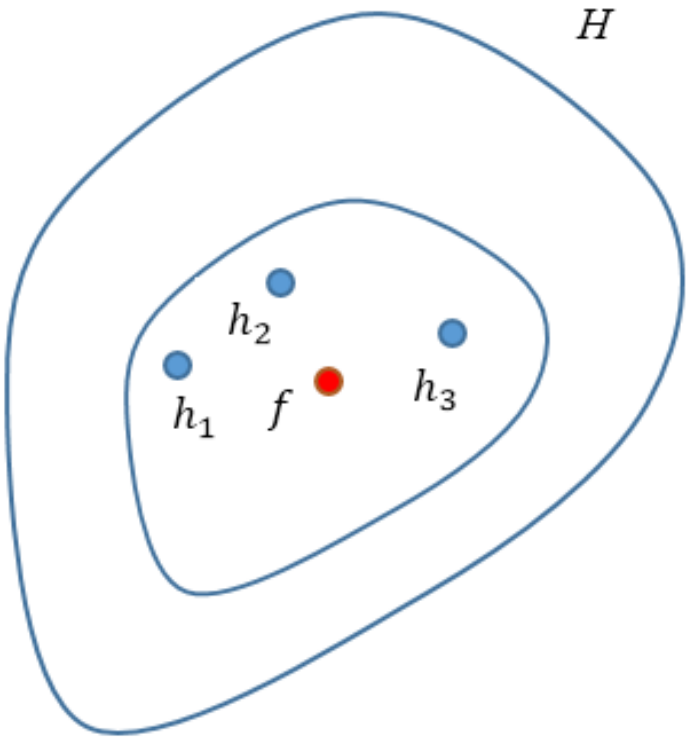

Figura 6 – Representação Estatística do Ensemble

A grande parte dos algoritmos de aprendizado de máquina operam mediante algum tipo de busca local, e isto pode gerar a solução de um problema mediante um ótimo local. Esses métodos tem como base a aplicação do algoritmo gradiente descendente para minimizar a função mediante os dados de treinamento; por outro lado existem os algoritmos baseados em árvores de decisão onde estes empregam uma regra de partição gulosa para realizar o crescimento da árvore. Portanto, a combinação dessas duas particularidades supracitadas dos algoritmos de aprendizado de máquina geram uma força maior no que diz respeito a forma de busca do método *ensemble*, pois podemos construir um conjunto de máquinas de aprendizado iniciando a busca local em diferentes pontos de partida, com

isto obteremos uma boa aproximação para a função f desconhecida o que fica um pouco difícil de se obter mediante o aplicação de apenas um regressor. Esta ideia é mostrada na figura 7.

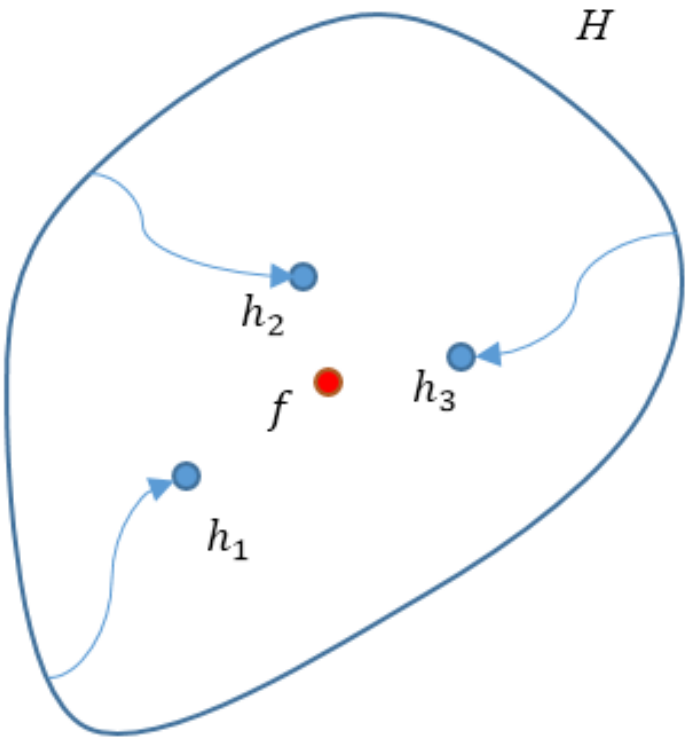

Figura 7 – Representação Computacional do Ensemble

Um outra característica do método ensemble resulta da sua possibilidade de representar uma função que não esta definida no espaço total de hipóteses H. Esta situação é uma situação que pode ser enfrentada em muitos problemas da vida real, uma vez que, nem sempre é possível estabelecer um espaço solução para o problema em análise. Para vencer essa dificuldade o método ensenble pode, através das hipóteses obtidas do espaço H, extender esse espaço representativo de funções de forma a definir a função desejada f (DIETTERICH, 2000). Isto pode ser representado intuitivamente mediante a figura 8.

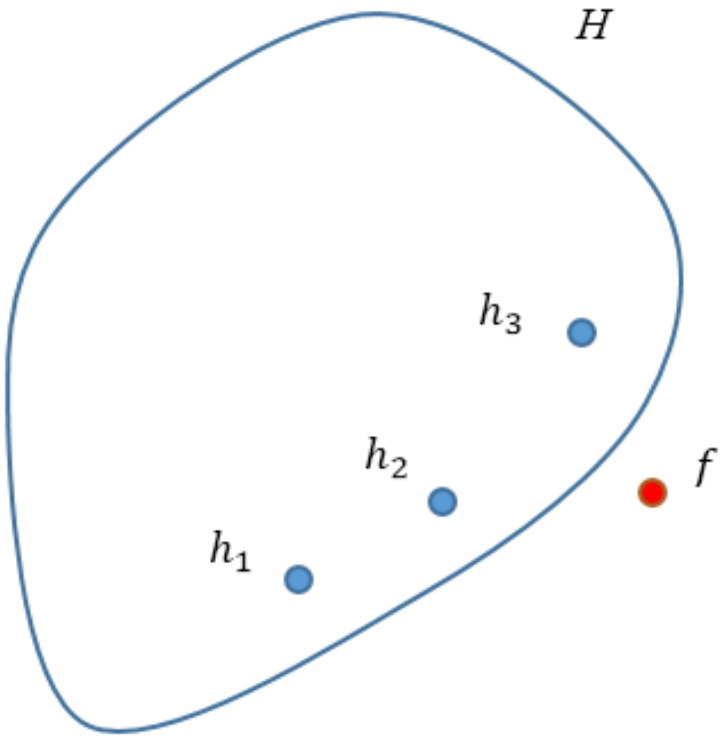

Figura 8 – Representação Computacional do Ensemble

Como é visto, os modelos ensemble apresentam melhor desempenho uma vez que trabalham juntos para melhorar o resultado final, e com base nisso foram aplicados alguns modelos que, com mais propriedade, conseguem representar melhor o comportamento da distribuição de minério no subsolo.

3.8.1 Método *AdaBoost*

O primeiro passo na operação desse algoritmo é a partição no treinamento dos dados. Nesta fase, a entrada para o AdaBoost (ABR) é um conjunto de exemplos na forma $S = (x_1, y_1), (x_2, y_2), ..., (x_m, y_m)$ onde:

1. Cada x_i representa um vetor do sistema de atributos.
2. Para o sistema analisado, cada y_i representa a *label* associada com o x_i.
3. O número total de exemplos de treinamento é igual a m.

O algoritmo trabalha como mostrado do diagrama da figura 9. Um regressor do tipo AdaBoost é um meta-estimador que começa adaptando um regressor sobre o conjunto de dados original e depois ajusta copias adicionais do regressor sobre o mesmo banco de dados, onde os pesos das instâncias são ajustados de acordo com a predição atual (DRUCKER, 1997). Para este caso específico foi utilizado o algoritmo AdaBoost.R2.

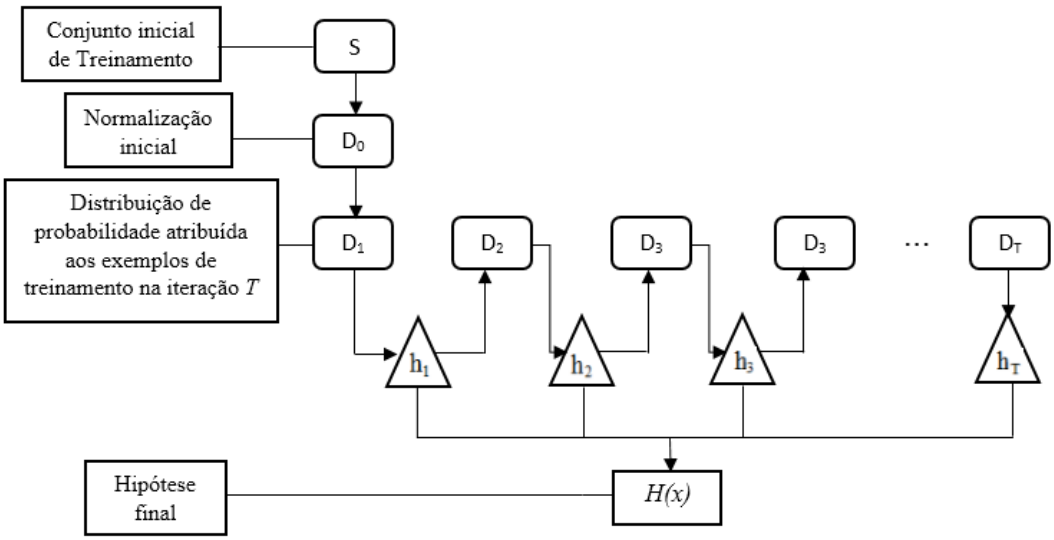

Figura 9 – Representação gráfica do AdaBoost

3.8.2 Gradiente Boosting

O Gradient Boosting, ou aumento de gradiente (GBR) é um aprimoramento da técnica boosting, esta tecnica tem como objetivo principal reduzir i *viés* e a *variância* do modelo final (BREIMAN, 1996). Esta metodologia lida com qualquer função de custo diferenciável. Esse é um procedimento preciso e efetivo que pode ser usado tanto para problemas de regressão quanto de classificação. Algumas vantagens deste método são:

1. Manuseio mais natural com dados de diferentes naturezas, (i.e. atributo ou característica heterogênea).
2. Acentuado poder preditivo.
3. Robustez concernente aos outliers no espaço de saída, por meio de funções de custo mais representativas, (FRIEDMAN, 2001).

Portanto, este modelo baseia-se na diminuição sucessiva da função custo, por meio do gradiente descendente. Para entender melhor como funciona, vamos considerar a sua formulação matemática como a seguir:

$$F(x) = \sum_{m=1}^{M} \gamma_m h_m(x) \tag{3.18}$$

Onde $h_m(x)$ são as funções básicas que são geralmente chamadas de aprendizes fracos no contexto da técnica boosting. O Gradiente Boosting usa árvores de decisão de tamanho fixo como aprendizes fracos, ou seja, as árvores de decisão possuem várias habilidades que as tornam valiosas para o aprimoramento, gerando uma maior capacidade de manipular dados do tipo misto e também a capacidade de modelar funções complexas. Assim, é um

tipo de modelo aditivo, isto é, ele constrói o modelo aditivo em um estágio progressivo, da seguinte forma:

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x) \tag{3.19}$$

Portanto, em cada estágio a árvore de decisão $h_m(x)$ é escolhida para minimizar o função de custo L dado o modelo atual F_{m-1} e seu ajustamento $F_{m-1}(x_i)$, tais como:

$$F_m(x) = F_{m-1}(x) + \arg\min_h \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + h(x)) \tag{3.20}$$

3.8.3 Florestas Aleatórias

Em Random Forest Regressor, ou regressor de floresta aleatória (RFR), cada árvore do conjunto de árvores é construída a partir de uma amostra sorteada com reposição, ou seja, uma amostragem do tipo bootstrap, dentro do conjunto de treinamento (BREIMAN, 2001). Além disso, ao dividir um *nó* durante a construção da árvore, a divisão escolhida não é mais a melhor divisão entre todos os conjuntos de atributos ou características. Em vez disso, a divisão escolhida é a melhor divisão entre um subconjunto aleatório dos atributos. Como resultado dessa aleatoriedade, o viés da floresta geralmente aumenta ligeiramente (com relação ao viés de uma única árvore não aleatória), mas, devido à média, sua variância também diminui, geralmente mais do que compensando o aumento do viés, daí resultando um modelo global melhor. Este método é, basicamente, um conjunto de $CART's$, uma vez que cria n árvores que usa o voto majoritário para classificação e a média para a regressão dos resultados, respectivamente. Assim, a Figura 10 apresenta um arranjo de uma floresta aleatória.

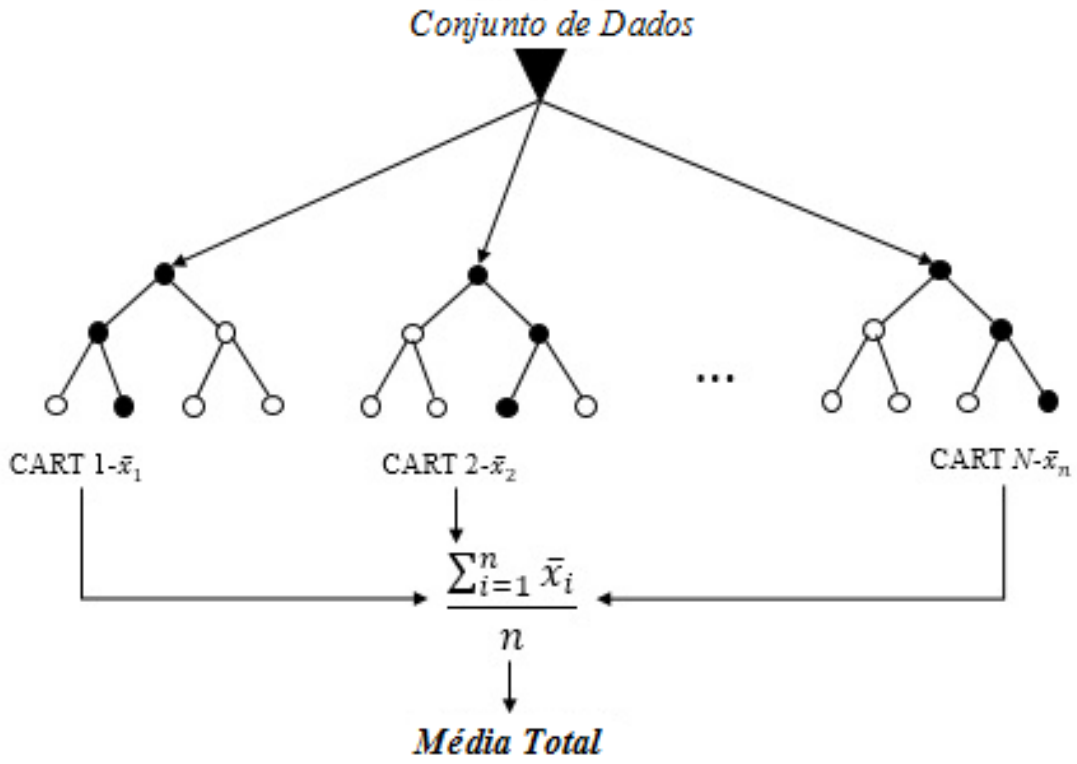

Figura 10 – Representação gráfica do Florestas Aleatórias

Portanto, o algoritmo floresta aleatórias, basicamente, constitui uma combinação de CART, uma vez que cada um deles realiza uma representação do conjunto de dados randomizado, o que permite avaliar o comportamento do subsolo a partir de diferentes pontos de vista. Isto configura-se uma aplicação bastante adequada para a estimativa de minério porque este método pode aprender diversas características do subsolo por diferentes perpectivas.

Essa forma de aprendizado se configura como uma forma adequada para o estudo em estimativa de recursos minerais porque esse método de AM minimiza a variância do erro a cada iteração, ou seja, em cada partição do banco de dados ele encontra o conjunto de dados cuja a variância entre eles é mínima. E isso se deve a seguinte equação de treinamento de dados:

$$I_v(N) = \frac{1}{|S|^2} \sum_{i \in s} \sum_{j \in s} \frac{1}{2} (x_i - x_j)^2 - \left(\frac{1}{|S_t|^2} \sum_{i \in s_t} \sum_{j \in s_t} \frac{1}{2} (x_i - x_j)^2 + \frac{1}{|S_f|^2} \sum_{i \in s_f} \sum_{j \in s_f} \frac{1}{2} (x_i - x_j)^2 \right) \quad (3.21)$$

Onde S é o conjunto de partição em análise e S_t e S_f são o conjunto de amostras para os quais o teste de partição é verdadeiro (*true*) ou falso (*false*) respectivamente. Essa equação pode ser interpretada como uma medida de similaridade entre os teores circunvizinhos o que se mostra como uma propriedade muito importante para análise de dados espaçados.

3.8.4 Árvores Extra-Aleatórias

Este método, Extremely Randomized Regressor, ou regressor extremamente randomizado (ERR), consiste, essencialmente, em randomizar fortemente ambos os atributos e a escolha de ponto de corte ao dividir um *nó* de árvore. Ele constrói árvores totalmente aleatórias cujas estruturas são independentes dos valores de saída da amostra de aprendizagem. A força da randomização pode ser ajustada às especificidades do problema pela escolha apropriada de um parâmetro. Como em florestas aleatórias, um subconjunto aleatório de atributos é usado, mas em vez de procurar os limites mais discriminativos, os limites são desenhados aleatoriamente para cada atributo e o melhor desses limites gerados aleatoriamente é selecionado como a regra de divisão. Isso geralmente permite reduzir um pouco mais a variância do modelo, em detrimento de um aumento ligeiramente maior do *viés* (GEURTS; ERNST; WEHENKEL, 2006)

O algoritmo Extra-Trees constrói um conjunto de árvores de decisão ou regressão não-ajustadas de acordo com o procedimento clássico de cima para baixo. Suas duas principais diferenças com outros métodos de agrupamento baseados em árvores são que ele divide os *nós* escolhendo pontos de corte totalmente aleatórios e que usa toda a amostra de aprendizado (em vez de uma réplica de bootstrap) para cultivar as árvores. Para ver como funciona um Regressor de árvores extra-aleatórias, observervemos o seguinte pseudocódigo:

Algoritmo 2: Pseudocódigo do Algoritmo Extra Trees

Construa um comitê de Extra Trees(S).
Input: um conjunto de treinamento S
Output: uma árvore de comitê $\tau = \{t_1, ..., t_M\}$.
for i=1 a M **do**
  Gerar uma árvore: t_i=**Construa uma Extra Trees** (S)
  return τ
end for
Construa uma Extra Trees (S).
Entrada: a training set S.
Saída: a tree t.
return uma folha rotulada pela classe de frequência (ou média de saída na regressão) em S
if $|S| < n_{min}$ **then**
  todos os atributos candidatos são constantes em S, ou
  a variável de saída é constante em S
else
  Selecione K atributos, $\{a_1, ..., a_K\}$, sem reposição, entre todos, não constante, em S atributos candidatos;
  Gerar K partições $\{s_1, ..., s_K\}$, onde s_i = **Pick a random split** (S, a_i), $\forall i$ = 1,...,K;
  Selecione uma partição s_* tal que Score (s_*, S)=$max_{i=1,...,K}$ Score (s_i, S);
  Partição S em subconjuntos S_l e S_r de acordo com o teste s_*;
  Construa t_l = **Construa uma Extra Tree** (S_l) and t_r = **Construa uma Extra Tree Construa uma Extra Tree** (S_r) from these subset;
  Crie um nó com a partição s_*, una-o t_l e t_r como esquerda e direita subárvores deste nó e retorne o resultado da árvore t.
end if
Selecione uma partição aleatória (S, a)
Entrada: um conjunto de treinamento S e um atributo a.
Output: uma partição.
if o atributo a é numérico **then**
  Calcule o máximo e o mínimo valor de a em S, denote respectivamente por a^s_{min} e a^s_{max};
  Desenhe um ponto de poda a_c uniformimente em $[a^s_{min}, a^s_{max}]$;
  return a partição $[a < a_c]$
end if
if o atributo a é categórico denotado por A **then**
  Calcule A_S o subconjunto de S de valores de a que aparecem em S;
  Aleatóriamente desenhe um subconjunto própio não vázio A_1 de A_S e um subconjunto A_2 de $A \backslash A_S$;
  return a partição $[a \in A_1 \cup A_2]$
end if

4 ANÁLISE EXPERIMENTAL

O estudo experimental está fundamentado na análise do comportamento estatístico dos dados realizado juntamente com a seleção de modelos de aprendizado de máquina. Esta abordagem foi estabelecida de forma que o todos os experimentos, com cada uma das metodologias aplicadas, pudessem gerar os diversos modelos geológicos de teores com o máximo perfeição.

4.1 AMBIENTE COMPUTACIONAL

Todos os códigos foram implementados na linguagem de programação Python utilizando-se da plataforma de implementação Jupyter, a qual possui uma ampla gama de bibliotecas para a elaboração de códigos utilizando os algoritmos de aprendizado de máquina.

4.2 BANCO DE DADOS

Usamos três conjuntos de dados de operações de mineração para validar nosso método. Cada conjunto de dados é composto por três atributos e um rótulo. O rótulo, alvo ou *label* significa nosso teor de minério, ou seja, representa a variável que é nosso objeto de inferência. Os três atributos, de cada conjunto de dados, são obtidos através de uma perfuração georreferenciada no maciço rochoso, isto é, os atributos são capturados definindo um ponto georreferenciado no plano X e Y e variando a profundidade Z. A variação da coordenada Z nos traz as informações sobre a quantidade do teor de minério que está no subsolo e a partir dessas informações é extraído o corpo de prova, que contém o respectivo teor de minério de acordo com suas profundidades. Essas profundidades são obtidas considerando o comportamento geológico de cada depósito de minério. Em nossa abordagem, analisamos um depósito de ferro Fe, um depósito de ouro Au e um depósito de níquel Ni.

4.3 SELEÇÃO DE MODELOS

A seleção de modelos é base fundamental para todos os experimentos com os algoritmnos de aprendizado de máquina, pois é através dessa seleção que é possível encontrar a melhor máquina de aprendizado que pode ser gerada para definir o comportamento dos dados experimentais.

Aqui serão apresentadas cada uma das metodologias implementadas para a resolução do problema de estimativas de recursos minerais.

4.3.1 Regressão Linear, Lasso e ElasticNet

O objetivo de aplicar esses três métodos lineares foi verificar se as instâncias que explicam o corpo de minério podem ser interpretadas por um modelo linear, ou seja, se elas podem explicar o comportamento interno da distribuição de teor do mineral, do corpo mineralizado, através de uma combinação linear destas variáveis dadas no banco de dados. Cada conjunto de dados foi dividido em 80% de treinamento e 20% de conjuntos de testes, a fim de otimizar esses métodos. Esta configuração da partição do conjunto de dados foi usada para todos os métodos deste trabalho. Os resultados para os métodos lineares são mostrados na tabela 2.

Legenda	
MI	Máximo de Iterações
T	Tolerância

Tabela 2 – Os melhores parâmetros do *ENet* e *Lasso*

	Fe	*Au*	*Ni*
MI	100	100	100
T	10^{-3}	10^{-3}	10^{-3}

Tabela 3 – Parâmetros inseridos no Grid de Busca do *ENet* e *Lasso*

MI	100	1000	10000	100000
T	10^{-3}	10^{-4}	10^{-5}	10^{-6}

Para o modelo *ElasticNet* e *Lasso*, usamos um grid de busca ou *grid search* de forma a obter o melhor conjunto de parâmetros que pudesse representar a melhor máquina de aprendizado para os dois modelos supracitados. Os parâmetros variados foram, *Número Máximo de Iterações* e *Tolerância* mostrados na tabela 3. Com os quais obtivemos os respectivos resultados encontrados para *ENet* e *Lasso* mostrados na tabela 18. De acordo com os melhores parâmetros, mostrados na tabela 2, podemos ver, através do resultados, que não há diferença entre esses dois métodos citados para os três conjuntos de dados estudados.

4.3.2 Árvore de Regressão

Como podemos ver, através dos resultados mostrados na tabela 19, o algoritmo CART é quase equivalente, em termos de resultados, a krigagem. Mas não superou a teoria clássica da geoestatística.

Legenda	
MAP	Número Mínimo de Amostra para Partição
MAF	Número Mínimo de Amostra para uma Folha
MA	Número Máximo de Atributos
FPP	Mínima Fração Ponderada dos Pesos

Tabela 4 – Parâmetros inseridos no Grid de Busca do *CART*

MAP	2	3	4	5	6	7	8
MAF	1	2	3	4	5	6	
MA	auto	sqrt	log2				
FPP	0	0.5					

Tabela 5 – Melhores parâmetros para o *CART*

	Fe	Au	Ni
MAP	8	3	7
MAF	3	5	3
MA	sqrt	log2	auto
FPP	0	0	0

4.3.3 Máquinas de Vetores de Suporte

Mesmo que represente um poderoso algoritmo para resolver uma ampla gama de problemas, podemos ver, por meio dos resultados mostrados na tabela 19, que não foi capaz de ser melhor que a teoria clássica.

Legenda	
C	Parâmetro de Penalidade do Erro
D	Grau do Polinômio da Função Kernel
T	Tolerância
K_e	Função Kernel

Tabela 6 – Parâmetros inseridos no Grid de Busca do *SVR*

C	2^{-6}	2^{-3}	2^{0}	2^{3}	2^{6}
D	1	2	3	4	
T	10^{-2}	10^{-3}	10^{-4}		
K_e	poly	rbf	sigmoid		

Tabela 7 – Melhores parâmetros para o SVR

	Fe	Au	Ni
C	1	64	64
D	1	1	1
T	10^{-4}	10^{-4}	10^{-3}
K_e	rbf	rbf	rbf

4.3.4 Vizinho mais Próximo

O método KNN foi um dos melhores métodos aplicados neste trabalho, pois superou, em todos os conjuntos de dados, o método tradicional para estimar o recurso mineral, conforme mostrado na tabela 19.

Legenda	
K	Número de Vizinhos Utilizados no Cálculo
A	Algoritmo Utilizado no Cálculo dos Vizinhos
DF	Dimensão das Folhas
P	Função de Peso Utilizada na Predição

Tabela 8 – Parâmetros inseridos no Grid de Busca do KNN

K	1	3	5	7	9	11
	13	15	17	19	21	
A	auto	ball tree	kd tree	brute		
DF	10	20	30			
P	uniform	distance				

Tabela 9 – Melhores parâmetros para o KNN

	Fe	Au	Ni
K	3	7	3
A	auto	ball tree	auto
DF	30	10	20
P	distance	distance	distance

4.3.5 Regressor AdaBoost

Este método, apesar de trabalhar razoavelmente rápido nos três conjuntos de dados, não superou os resultados obtidos pelo método da krigagem.

Legenda	
TA	Taxa de Aprendizado
NE	Número de Estimadores

Tabela 10 – Parâmetros inseridos no Grid de Busca do ABR

TA	10^{-3}	10^{-2}	10^{-1}	10^{0}	10^{1}	10^{2}	10^{3}
NE	10	20	30	40	50		
	60	70	80	90	100		

Tabela 11 – Melhores parâmetros para o ABR

	Fe	Au	Ni
TA	1	10^3	10^3
NE	20	10	90

4.3.6 Gradiente Boosting

O Gradient Boosting desempenhou um papel importante nos conjuntos de dados para o Fe e o Au, mas falhou no Ni. Apesar de não funcionar com o conjunto de dados do Níquel, podemos ver através dos resultados, mostrados na tabela 20, que este método conjunto funcionou perfeitamente para mapear o comportamento do Ferro e do Ouro no subsolo.

Legenda	
NE	Número de Estimadores
MAP	Número Mínimo de Amostra para Partição
MAF	Número Mínimo de Amostra para uma Folha
TA	Taxa de Aprendizado
SA	Sub Amostragem

Tabela 12 – Parâmetros inseridos no Grid de Busca do GBR

NE	50	60	70	80	90	100	110
MAP	2	3	4	5	6	7	8
MAF	1	2	3	4	5	6	
TA	0.1	0.2	0.3	0.4	0.5		
SA	0.1	0.2	0.5	1			

Tabela 13 – Melhores parâmetros para o GBR

	Fe	Au	Ni
NE	90	50	110
MAP	3	5	5
MAF	4	1	1
TA	0.3	0.5	0.5
SA	1	1	1

4.3.7 Florestas Aleatórias

A floresta aleatória funcionou perfeitamente em todos os conjuntos de dados, ou seja, reconheceu o padrão de distribuição dos três minerais no subsolo. De acordo com os resultados mostrados na tabela 20, vê-se que o algoritmo de florestas aleatórias superou o método tradicional de estimativa de recursos minerais.

Legenda	
NE	Número de Estimadores
MAP	Número Mínimo de Amostra para Partição
MAF	Número Mínimo de Amostra para uma Folha
MA	Número Máximo de Atributos

Tabela 14 – Parâmetros inseridos no Grid de Busca do RFR

NE	50	60	70	80	90	100	110
MAP	2	3	4	5	6	7	8
MAF	1	2	3	4	5	6	
MA	auto	sqrt	log2				

Tabela 15 – Melhores parâmetros para o RFR

	Fe	Au	Ni
NE	90	100	110
MAP	2	4	2
MAF	1	1	1
MA	auto	sqrt	log2

4.3.8 Árvores Extra-Aleatórias

Este método foi o melhor de todos os métodos aplicados para a estimativa de recursos minerais em subsolo. Este método desempenhou um papel excelente em todos os conjuntos

de dados e teve um tempo de treinamento relativamente curto para aprender o comportamento da distribuição de minerais no subsolo. De acordo com os resultados, podemos ver que seu desempenho foi satisfatoriamente melhor que todos os métodos utilizados nesta pesquisa.

Legenda	
NE	Número de Estimadores
MAP	Número Mínimo de Amostra para Partição
MAF	Número Mínimo de Amostra para uma Folha
MA	Número Máximo de Atributos

Tabela 16 – Parâmetros inseridos no Grid de Busca do ERR

NE	50	60	70	80	90	100	110
MAP	2	3	4	5	6	7	8
MAF	1	2	3	4	5	6	
MA	auto	sqrt	log2				

Tabela 17 – Melhores parâmetros para o ERR

	Fe	Au	Ni
NE	90	50	100
MAP	2	3	2
MAF	1	1	1
MA	auto	auto	auto

5 RESULTADOS E DISCUSSÃO

De acordo com os resultados obtidos por meio da aplicação dos métodos de AM: *LR*, *ENet* e *Lasso*, pode-se observar que eles não atingiram o objetivo pretendido, que era, inicialmente, encontrar um erro de estimativa para os recursos mineriais inferior ao encontrado pelo método da krigagem, o que nos mostra, a princípio, que esses métodos lineares não são eficazes no que concerne ao astudo de estimativas de recursos minerais.

Tabela 18 – MSE e STD para as abordagens de *Krig LR*, *Lasso* e *ENet*

	Fe		Au		Ni	
Krig	0.003531	0.000632	0.69987	0.49704	0.11489	0.029237
LR	0.004250	0.000502	0.788360	0.349094	0.219266	0.019958
Lasso	0.004409	0.000556	0.787767	0.351129	0.249742	0.021609
ENet	0.004409	0.000556	0.787767	0.351129	0.249742	0.021609
	MSE	STD	MSE	STD	MSE	STD

Em uma análise feita com os algoritmos *CART*, *SVR* e *KNN*, evidenciou-se que, conforme os resultados expressos na tabela 19, o *KNN* foi o que apresentou o melhor desempenho para os bancos de dados de ferro, ouro e níquel, em comparação com o método tradicional da krigagem.

Tabela 19 – MSE e STD para as abordagens de *Krig*, *CART*, *SVR* e *KNN*

	Fe		Au		Ni	
Krig	0.003531	0.000632	0.69987	0.49704	0.11489	0.029237
CART	0.003546	0.00028	0.737443	0.244427	0.144253	0.007293
SVR	0.003414	0.000376	0.798644	0.349169	0.210716	0.020379
KNN	0.002491	0.000497	0.562367	0.174921	0.090542	0.004007
	MSE	STD	MSE	STD	MSE	STD

Os métodos de AM que apresentaram uma melhor performance na estimativa de teores para os minérios de ferro, ouro e níquel foram os de comitê ou conjuntos, também chamados de ensembles.

A abordagem mais eficiente, no que se refere à estimativa de recursos minerais com aplicações de metodologias ensemble, conforme tabela 20, se deu por meio da aplicação do Regressor Extremamente Randomizado (expressão oriunda da sigla *ERR*, em inglês).

Tabela 20 – MSE e STD para as abordagens de *Krig*, *ABR*, *GBR*, *RFR* e *ERR*

	Fe		Au		Ni	
Krig	0.003531	0.000632	0.69987	0.49704	0.11489	0.029237
ABR	0.003604	0.000485	0.756158	0.325596	0.215055	0.019080
GBR	0.002856	0.000435	0.635008	0.218908	0.158813	0.014804
RFR	0.002433	0.000395	0.590905	0.226531	0.096966	0.008055
ERR	0.002293	0.000403	0.541706	0.169246	0.087621	0.005584
	MSE	STD	MSE	STD	MSE	STD

5.1 REPRESENTAÇÃO GEOLÓGICA

A seguir são mostrados os modelos geológicos de teores gerados por meio de todos os métodos de regressão em AM aplicados neste trabalho científico, incluindo a krigagem que se trata do método geoestatístico tradicional para a análise de estimativa de recursos minerais.

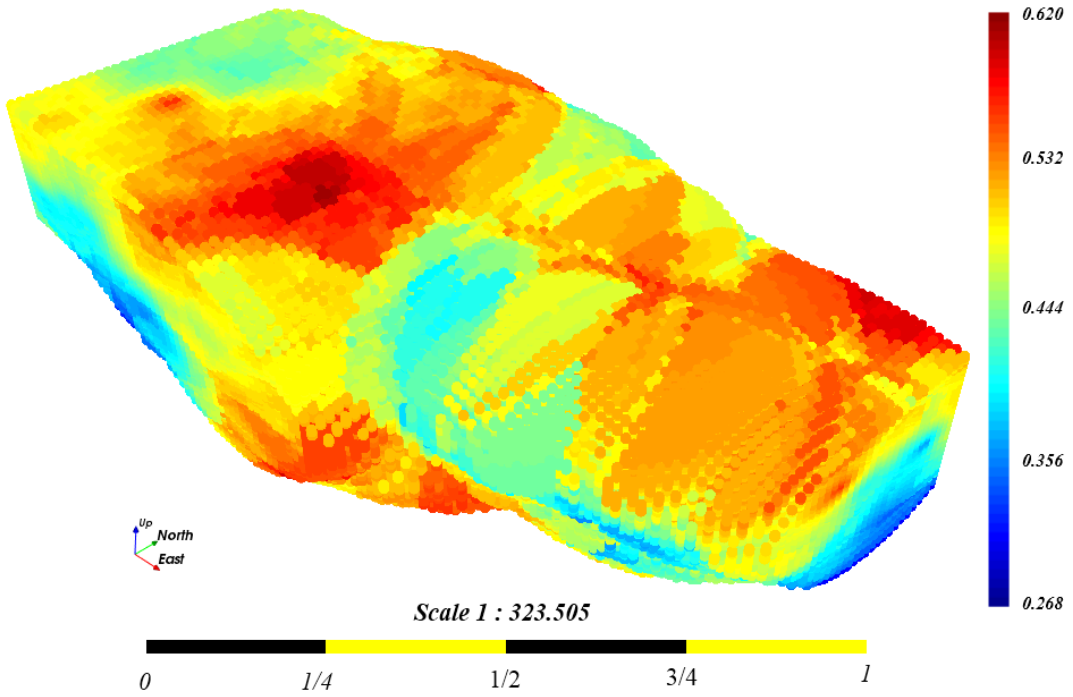

Figura 11 – Modelo de teores de ferro estimado pela krigagem

O modelo geológico de teores apresentado acima mostra a distrtibuição espacial do minério de ferro ao longo de toda a jazida mineral. A escala vertical de cores apresenta o nível de pureza das respectivas regiões que contêm o minério de ferro. Por exemplo, se iniciarmos um extração mineral nas regiões que estão representadas pelas cores laranja e vermelha, encontraremos, para cada tonelada de minério extraída, um valor que pode variar de uma quantidade de 532 a 620 kilogramas de minério de ferro, o que corresponde a um percentual de 53,2% a 62,0%, respectivamente. A partir dessa observação,

é possível afirmar que a distribuição dos teores ocorre de forma gradativa dentro corpo mineralizado, observando-se, às vezes, fortes variações de uma região para outra. Isto se deve ao fato de o algoritmo de krigagem ordinária apresentar uma váriância mínima em seu método de interpolação, o que, automaticamente, gera bons resultados em análise de dados espaçados.

No entanto, apesar do bom desempenho evidenciado pela propriedade da supracitada variância mínima, o algoritmo de krigagem assume funções analíticas (ou variogramas) muito acentuadas acerca da distribuição espacial dos teores, bem como do comportamento destes no subsolo. Portanto, pode-se afirmar que os variogramas acabam, de alguma forma, comprometendo a qualidade do modelo de teores gerado pela krigagem. Em outras palavras: é importante afirmar que há uma certa imprevisibilidade na adaptação das jazidas minerais às hipóteses variográficas existentes na literatura, ou seja, não é possível definir o comportamento da distribuição de teores em jazidas com alta descontinuidade.

Por utilizar funções variográficas preestabelecidas, o método de krigagem, em certos casos, impossibilita a realização da estimativa de recursos minerais em virtude do comportamento errático de algumas jazidas minerais. Entretanto, apesar das suas limitações, o algoritmo de krigagem pode realizar estimativas de recursos minerais com certo grau de precisão, devido à sua propriedade de variância mínima, a qual o torna um método de grande aplicabilidade para dados espaçados, ou seja, aqueles provenientes de sondagens de jazidas minerais.

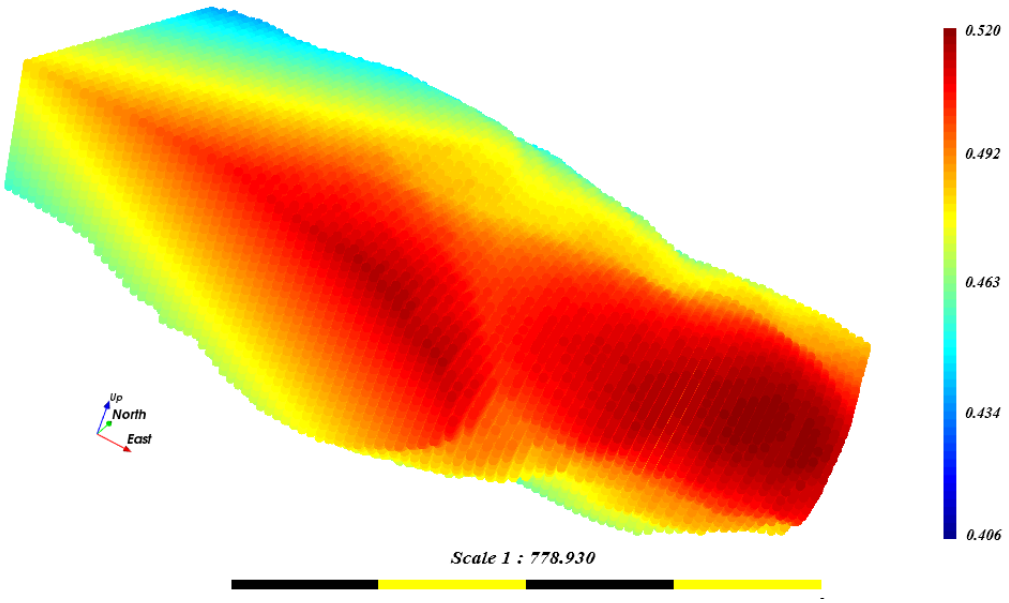

Figura 12 – Modelo de teores de ferro estimado pela regressão linear

A figura acima representa um modelo geológico de teores para o minério de ferro, realizado mediante o treinamento dos dados pelo algoritmo de regressão linear. A escala de cor vertical representa o percentual de teor (ou pureza) do minério de ferro retratado na imagem em destaque, de acordo com o modelo de regressão linear.

Comparando-se as figuras 11 e 12, podemos ver que há uma diferença bastante significativa, no que se refere à distribuição espacial das intensidades de teores. Isso se deve ao fato de que o modelo de regressão linear apresenta um viés muito forte, ou seja, ele tenta interpretar a distribuição de teores na jazida mineral como sendo uma reta. O mesmo não ocorre com a geologia das jazidas minerais. Em outras palavras, pode-se afirmar que não é possível descrever um fenômeno geológico mineralizado por meio do método da regressão linear, visto que o erro associado às estimativas de teores, conforme tabela 18, aumenta em 20,4%, comparado ao método tradicional de krigagem, o que configura como um método não adequado para estimativas de recursos minerias.

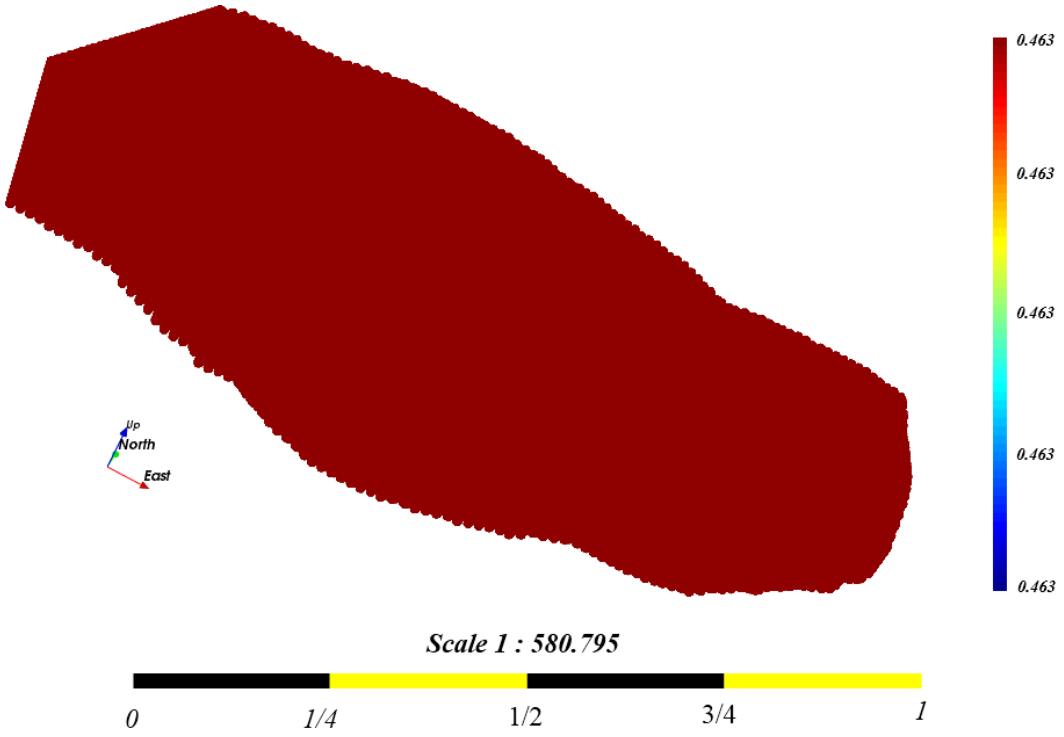

Figura 13 – Modelo de teores de ferro estimado pela regressão lasso

A figura 13 representa o modelo geológico de teores para o minério de ferro. Pode-se, vizualizar por meio da ilustração supracitada que o modelo geológico de teores gerado pelo treinamento dos dados de sondagem relacionados ao ferro, através do método de regressão *lasso*, apresenta um gradiente constante para a distribuição de teores no corpo mineralizado. Essa constância na representação da disposição de teores é motivada pela restrição algorítmica do modelo *lasso*. A referida restrição algorítmica, segundo se observa no modelo matemático 3.10, torna as variáveis independentes muito insensíveis a pequenas variações na análise do teor. Desta forma as variáveis explicativas as quais são representadas pelas coordenadas geográficas X, Y e Z não são relevantes o suficiente para explicar a variação na distribuição espacial de minério de ferro em todo o corpo mineralizado. A tabela 18 mostra que esse método apontou 25% de erro a mais na estimativa de

minério de ferro, se comparado com o método da krigagem, o que sugere não se tratar de um método apropriado para avaliação de recursos minerais.

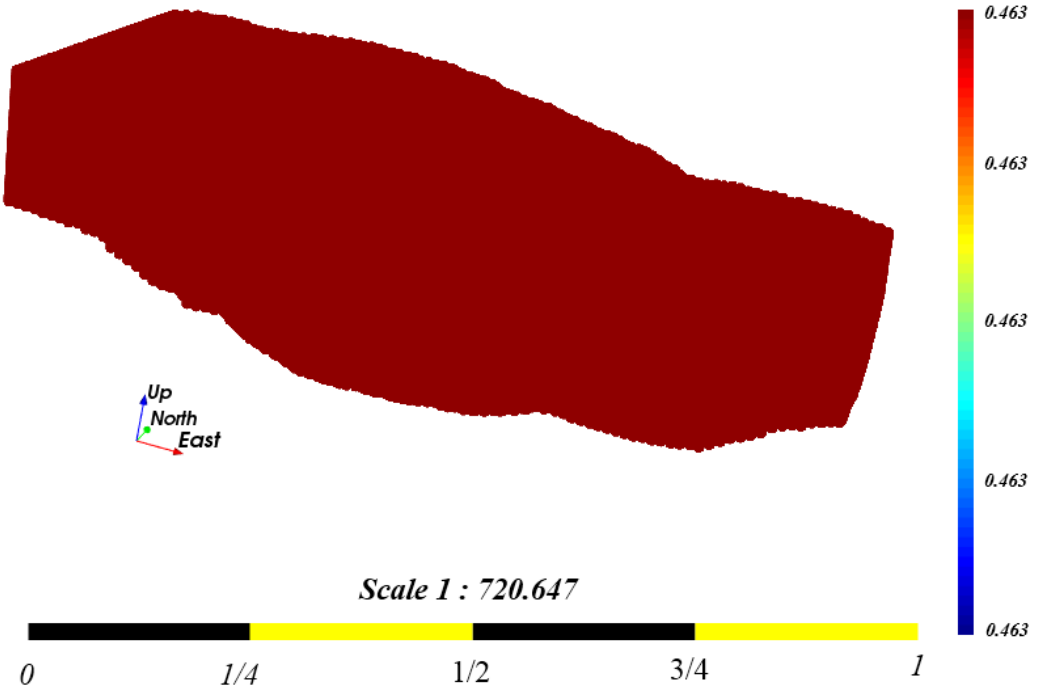

Figura 14 – Modelo de teores de ferro estimado pela regressão EN

A figura 14 mostra o modelo geológico de teores para o minério de ferro com a utilização da metogologia de aprendizado de máquina EN. Com base na imagem em destaque, observa-se que o referido método apresenta as mesmas características do *Lasso*. Isto se dá devido ao fator de regulalização dado pela restrição retratada no modelo 3.12, o qual é responsável por diminuir a sensibilidade das variáveis explicativas mediante pequenas alterações da variável alvo. O resultado gerado, mediante o treinamento dos dados de sondagem do minério de ferro, aponta um erro superior ao do método de krigagem, na casa de 25%, (tabela 18). Portanto, assegura-se que os modelos lineares não são bons representantes para a elaboração de modelos geológicos de teores para minérios de ferro.

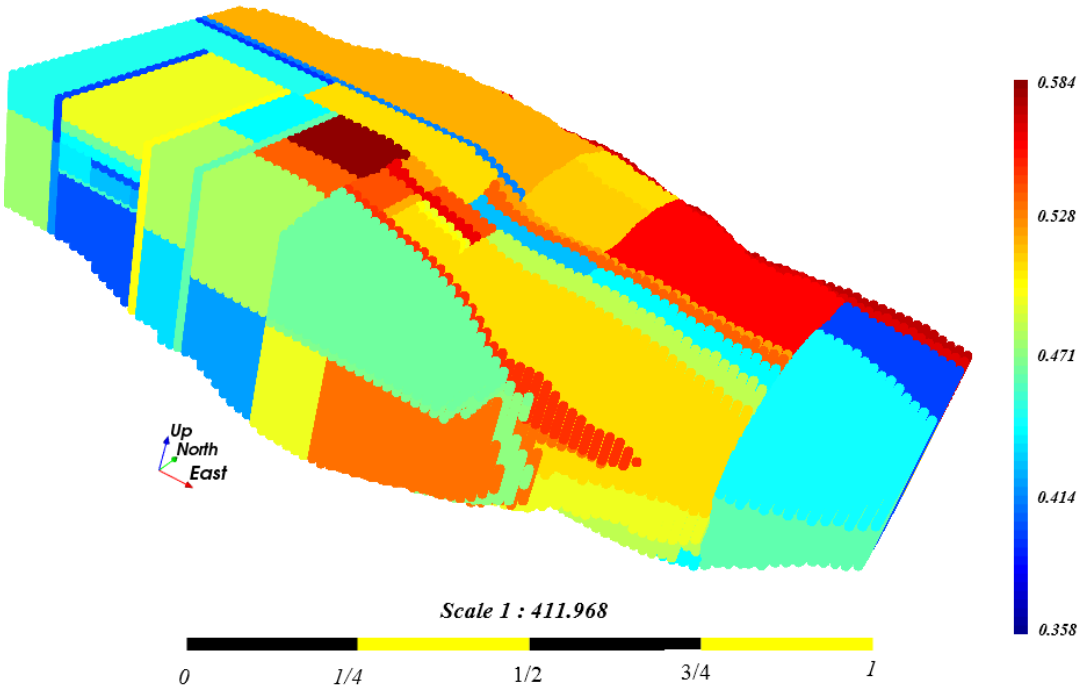

Figura 15 – Modelo de teores de ferro estimado pelo CART

A figura 15 nos mostra o modelo geológico de teores para o ferro. Neste modelo a escala de cores apresenta o percentual de pureza ditribuída ao longo de todo o corpo mineralizado. Este grau de pureza refer-se ao grau ou nível ferrífero encontrado nas diferentes regiões dadas pelas suas respectivas tonalidades de cores. Por exemplo se escolhermos extrair minério da região compreedida entre azul escuro e ezul claro encontraremos as quantidades de 358kg a 414kg respectivamente. O que corresponde a dizer que temos um percentual de pureza da ordem de 35,8% até 41,4%. Esse método de AM apresenta características muito importantes para a avaliação de recursos minerais porque ele baseia-se na construção de uma árvore de regressão que tem por finalidade encontrar a melhor medida de similaridade entre as distribuições de teores ao longo do corpo mineralizado. Esta medida de similaridade é dada pela equação 3.21. Sendo o algoritmo de árvore de regressão de natureza recursiva, esta equação nos permite avaliar a partição que possui maior similaridade entre teores a cada *nó* de crescimento da árvore, o que gera um conjunto de partições otimizadas para cada uma das regiões coloridas. Porém, mesmo com essa importante propriedade particional recursiva, o método $CART$ não se apresenta como um método adequado o suficiente para se realizar estimativas de recursos minerais. Pois, se contrapusermos as imagens 11 e 15 verificamos que existem semelhanças entre as regiões, mas a figura 15 apresenta uma variação muito intensa de uma região para a outra se comparada com figura 11. Isto é devido a uma característica particular do algoritmo de árvore de regressão a qual é: a baixa capacidade que esse método possui em diminuir a variância do erro das diferentes regiões, isto ocorre porque este modelo de AM trabalha apenas com o treinamento de uma árvore de regressão e por esta razão não possui baixa capacidade de

diminuir a variância do erro para o modelo matemático finalizado. Portanto, mesmo com uma boa capacidade adaptativa e interpretativa para dados espaçados vemos que esse método apresenta um ponto que não contribue para uma estimativa melhorada da jazida de minério de ferro como é mostrado mediante o resultado que se encontra na tabela 19 vemos que o esse método obteve um erro superior em 0,4% o que mostra que ele foi muito próximo do obtido com a krigagem.

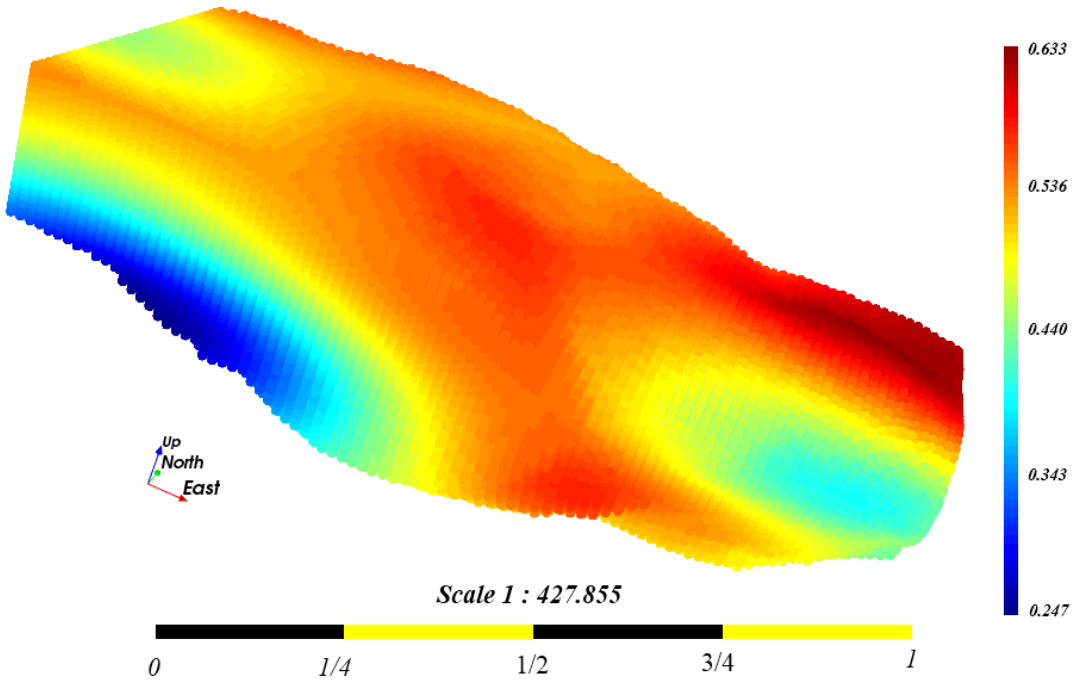

Figura 16 – Modelo de teores de ferro estimado pelo SVR

A figura 16 nos mostra o modelo geológico de teores para o minério de ferro realizado madiante a aplicação do método de aprendizado de máquina SVR. Este método de AM apresenta características importantes no que concerne a predição de dados espaçados devido a sua capacidade de classificar conjuntos de dados que possuem características tanto lineares quanto não lineares. Isto se deve ao fato desse método possuir um conjunto de funções analíticas ou função kernel K tabela 6 onde essas funções realizam tranformações algébricas e conseguem detectar as variações no padrão da variável de busca, que em nosso caso é teor de minério. Podemos entender como o padrão da variável de busca, neste contexto, como sendo regiões que possuem características semelhantes no que se refere a variabilidade de distribuição de teor dentro do corpo mineralizado. Se analisarmos as imagens 11 e 16 verificamos que existe uma diferença de estimativa de teores bastante significativa ao longo grande parte do corpo mineralizado. Contudo a sua apacidade analítica em avaliar as mínimas diferenças de teores dentro jazida mineral é bastante alta, e isto é um fator importante que o torna um método bastante robusto para a análise de dados espaçados. O maior problema consiste no conjunto de hipóteses que este método

de AM possui, entendamos como hipóteses o conjunto de funções analíticas que o método possui. Contudo, este método conseguiu um erro inferior ao método de krigagem da ordem de 3,3%. Portanto, esse método atende ao requisito de suavidade ou variância na interpretação dos dados espaçados, porém as suas hipóteses, dentro desse contexto da estimativa de recursos minerais, não contempla uma boa estratégia para estimativas de recursos minerais.

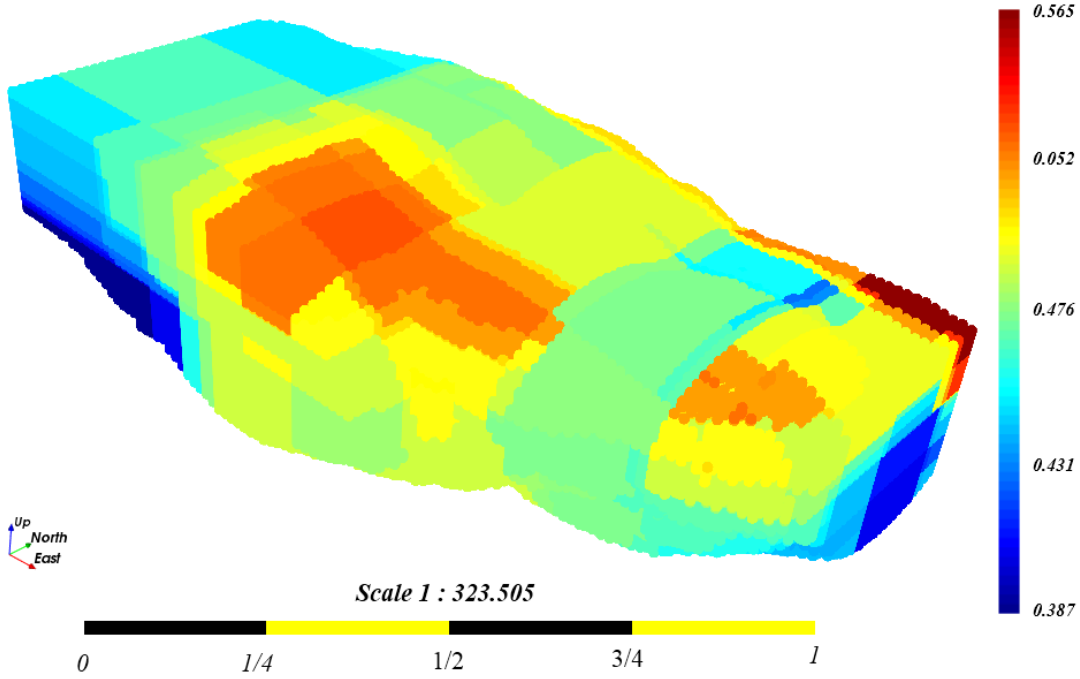

Figura 17 – Modelo de teores de ferro estimado por AdaBoost

A figura 17 apresenta o modelo de estimativa de teores com a utilização do método de AM *AdaBoost*. Ao visualizarmos as imagens 17 e 11 podemos observar que existem diferenças preditórias, ou seja, de estimativas entre as regiões do corpo mineralizado. Apesar desse algoritmo trabalhar com um conjunto de regressores e combiná-los de forma a obter uma menor variância para o erro de estimativa, ele apresenta características que o tornam com um viés menos flexível. Essas características remontam o conceito da incapacidade de funções analíticas descreverem, com maior precisão, o comportamento da distribuição espacial de teores em uma jazida mineral. Pois, isto acontece com esse método porque ele usa para o treinamento da máquina de aprendizado três funções analíticas que são: *linear*, *quadrática* e *exponencial*. Com isto podemos observar que existe uma semelhança entre o método de krigagem o método de AM *AdaBoost* porém a principal diferença entre os dois está no quesito de combinação de regressores o que não aconteçe com a krigagem. Vemos, através dos resultados mostrados na tabela 20 que esse método consegue até interpretar de forma satisfatória a distribuição de teores ao longo da jazida, ao apresentar um erro superior de 2% apenas se comparado ao método de krigagem. A sua falha é exatamente no quesito de treinamento dos dados espaçados, o que nos mostra de uma forma clara que

o comportamento anômalo de uma jazida muitas vezes não dá para ser interpretado por meio de modelos analíticos de funções matemáticas.

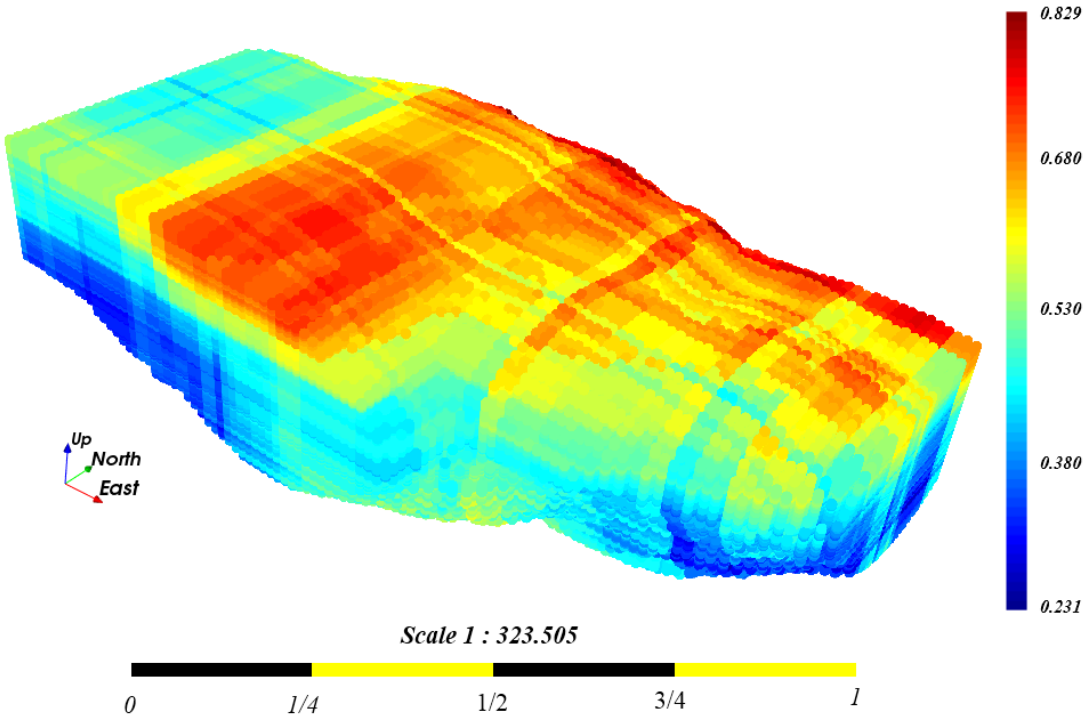

Figura 18 – Modelo de teores de ferro estimado por gradiente boosting

A figura 18 nos mostra modelo de teores gerado por meio da aplicação do algoritmo de AM Gradient Boosting. Se contrastarmos as figuras 18 com 11 veremos que existem semelhanças nas configurações de cores ao longo do corpo mineralizado. Este método, mesmo tendo características de treinamento parecidas com o *AdaBoost*, se mostra mais eficiente para a avaliação de dados espaçados. Isto se deve a forma de treinamento desse algoritmo visto que ele utiliza o método de gradiente descendente para conseguir encontrar os modelos otimizados, ou seja, encontrar o conjunto de regressores que melhor definem o comportamento do fenômeno estudado, que neste caso específico representa a distribuição de teores ao longo de uma jazida de minério de ferro. Através da tabela 20 vemos que esse método obteve um erro inferior ao da krigagem da ordem de 19% o que se mostrou bastante eficiente para os estudos de estimativas de recursos. Mas esta forma de treinamento ainda que se mostre adequada para o estudo de estimativa de recursos minerais, ela apresenta um problema que está na dificuldade de o gradiente descendente encontrar o ótimo global para cada uma das hipótese ou regressores. Portanto, ainda que seja um método viável para a análise de recursos minerais ele falha nesse quesito do gradiente descendente.

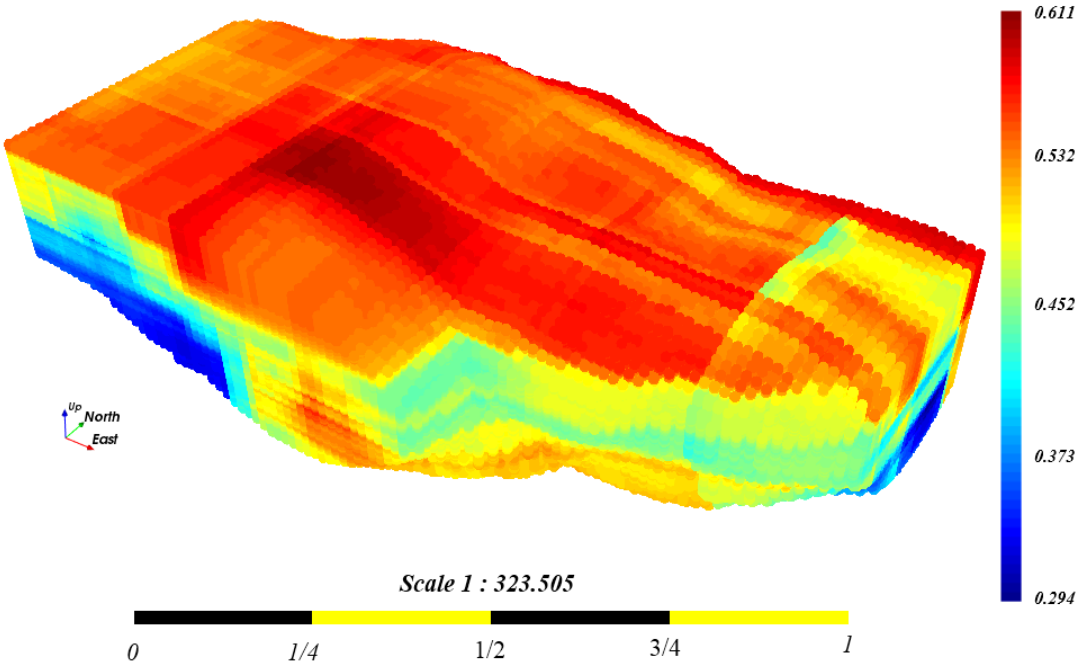

Figura 19 – Modelo de teores de ferro estimado por florestas aleatórias

A figura 19 mostra o modelo geológico de teores para o minério de ferro realizado com a utilização do método de AM florestas aleatórias ou simplesmente chamado de *random forest*. Ao compararmos as figuras 11 com a figura 19 verificamos que existem algumas diferenças no que concerne a distribuição espacial de teores no corpo mineralizado. Porém, observamos também que há uma semelhança muito forte na localização do grau máximo de teor do minério de ferro, porque as duas imagens apontam que o grau máximo de teor é encontrado na mesma região para os dois modelos, em outras palavras podemos dizer que as duas metodologias possuem algo em comum e este algo em comum é a capacidade que os algoritmos de krigagem e floresta aleatórias possuem de diminuir a variância do erro na estimativa de dados espaçados. Porém, uma vantagem bastante significativa que o método floresta aleatórias apresenta sobre o método da krigagem é o fato dele não fazer nenhuma hipótese acerca da distribuição espacial dos dados, ou seja, ele não condiciona o treinamento a nenhuma função analítica, o que se mostra como um método muito bom para a análise de estimativa de recursos minerais e isto é fefletido nos resultados mostrados na tabela 20 onde foi obtido um erro inferior ao da krigagem da ordem de 31%. O sucesso do método florestas aleatórias se dá exatamente porque ele consegue combinar os diversos regressores, que neste caso é um conjunto de árvores de regressão, com a sua capacidade de otimizar as partições de conjuntos de dados para cada *nó* mediante a aquação 3.21. Portanto, esse método nos fornece um novo olhar ou olhar melhorado no campo de análise de recursos minerais visto que ele se aprensenta como uma técnica bastante robusta para o estudo de dados espaçados.

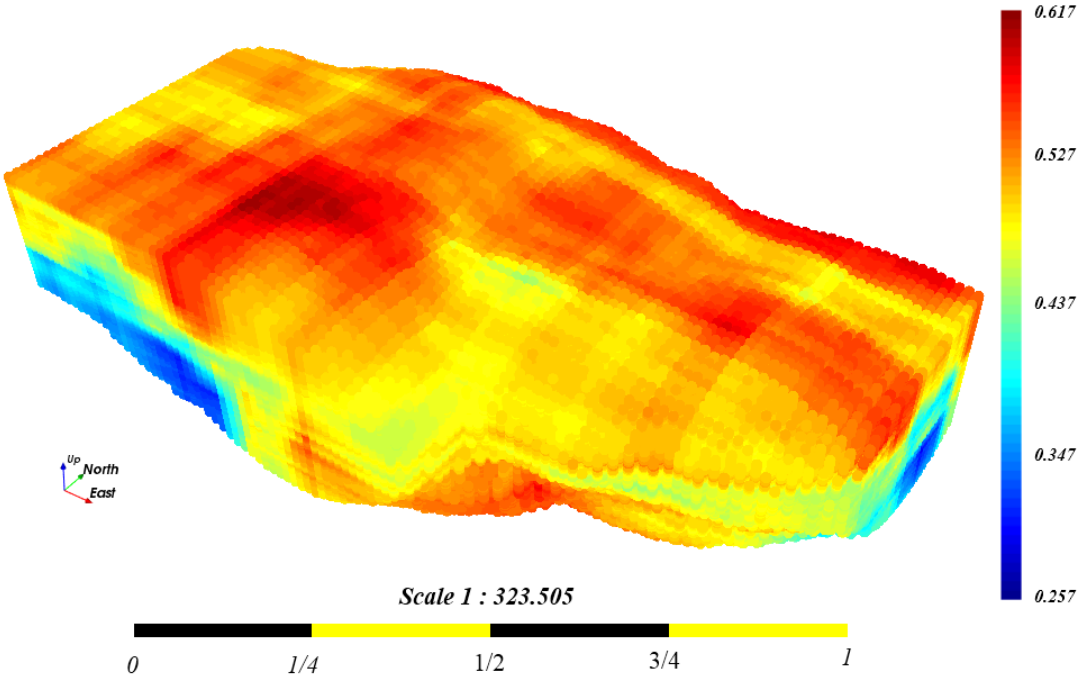

Figura 20 – Modelo de teores de ferro estimado por florestas extra aleatórias

A figura 20 mostra o modelo geológico de teores para o minério de ferro realizado com a aplicação do método de AM florestas extra aleatórias, o qual é simplesmente chamado de *Extra Trees*. Fazendo-se uma análise comparativa das imagens 11 e 20 observamos algumas variações para a distribuição de teores ao longo do corpo mineralizado. Contudo, se contrapusermos as imagens geradas pelos métodos de krigagem, florestas aleatórias e florestas extra aleatórias 11, 19 e 20 respectivamente, observaremos que existe o mesmo ponto comum que é a semelhança de localização de região que contém o maior teor de minério. Isto se confirma pela propriedade de minimização da variância do erro que ocorre nos três métodos supracitados.

O método florestas extra aleatórias apresenta uma característica que o torna um método ainda melhor que o de florestas aleatórias, essa característica é a sua forma de treinamanto dos dados, pois esta metodologia de AM trata cada conjunto de dados para treinamento de uma forma completamemte aleatória o que fornece uma flexibilidade ainda mais alta para a máquina de aprendizado finalizada ou modelo de estimativa final. Esta propriedade faz com que esse método seja capaz de trabalhar com jazidas com uma alta descontinuidade ou erráticas. Isto é confirmado mediante a diminuição do erro em 35% se comparado com o método tradicional. Portanto este método é perfeitamente aplicável para a análise de recursos minerais sejam eles contínuos ou descontínuos.

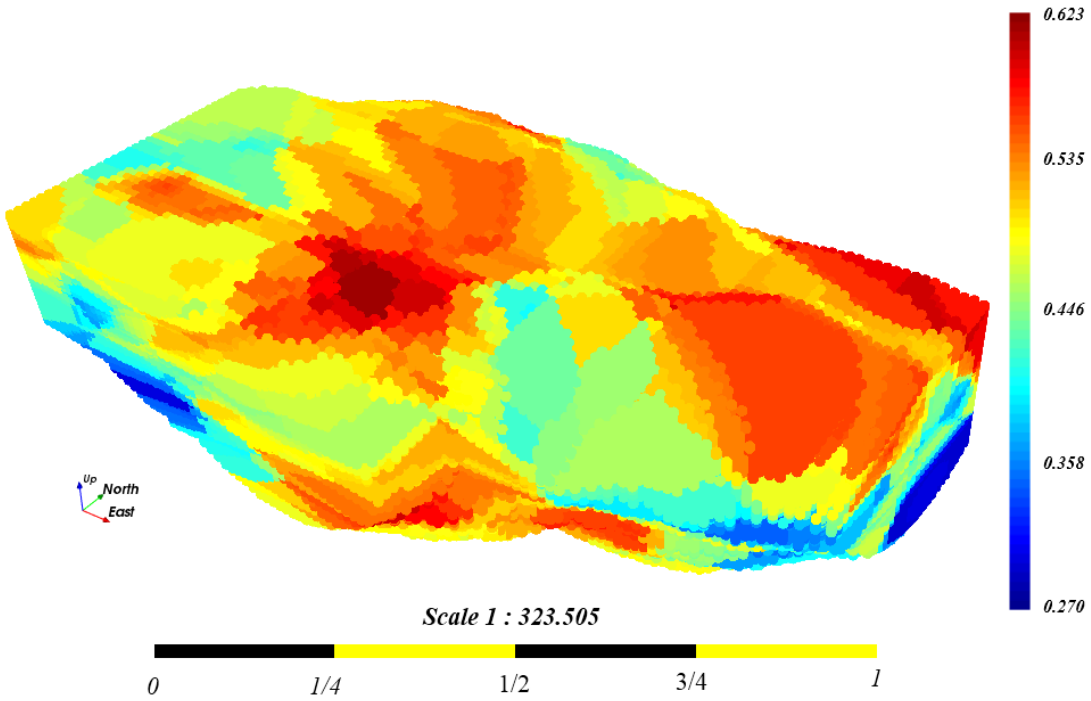

Figura 21 – Modelo de teores de ferro estimado por vizinho mais próximo

A figura 21 mostra o modelo geológico de teores para o minério de ferro obtido mediente a aplicação do método de AM KNN o qual é mais conhecido como os K vizinhos mais próximos. O KNN é um método relativamente simples onde cada ponto é estimado em termos dos teores dos pontos mais próximos ou vizinhos mais próximos. Conparando-se a imagen 11 com a imagem 21 vemos que há regiões semelhantes ao longo do corpo mineralizado. O fato de um algoritmo relativamente simples apresentar um erro de estimativa menor que o encontrado no método tradicional de krigagem está no fato de utilizarmos vários algoritmos para o cálculo do vizinho mais próximo, pois no treinamento dessa máquina de aprendizado vemos que o algoritmo *auto* se mostrou melhor para mapear os dados de ferro e níquel ao passo que o *ball tree* foi mais bem aplicado para o ouro, como mostrado na tabela 9. Portanto, observa-se que existe uma forte correlação espacial entre os teores de minério de ferro e esta correlação é tanto mais intensa quanto mais próximo um vizinho está do outro. Esse método apresentou uma diminuição no erro de estimativa da ordem de 29%. Configurando-se como uma alternativa aceitável para estimativa de recursos minerais.

Partindo da premissa de que existe uma forte correlação espacial entre a distribuição de teores dentro de um corpo mineralizado, fica claro que para realizar uma estimativa mais precisa devemos nos concentrar em um método que atenda esse quesito de correlacionalidade entre os dados sem considerar hipóteses fortes sobre a jazida mineral. Portanto, esse método de AM serve como um referencial para avaliar características semelhantes em outros métodos de AM para aplicação em estimativa de recursos minerais.

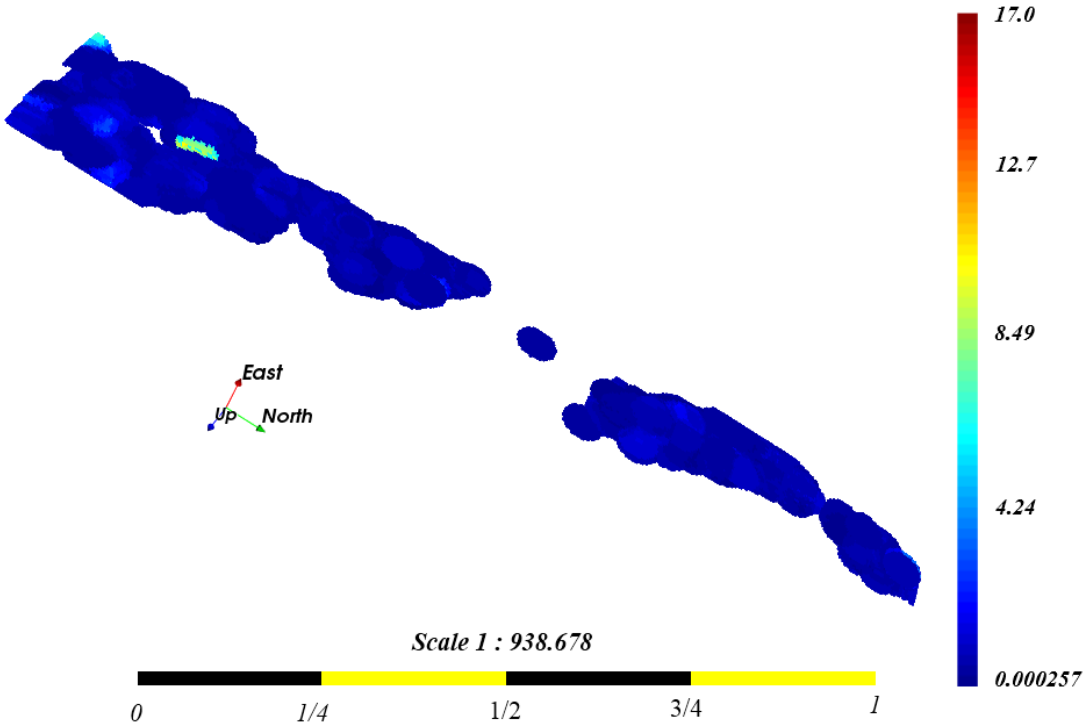

Figura 22 – Modelo de teores de ouro estimado pela krigagem

O modelo geológico de teores apresentado acima apresenta a distrtibuição espacial do minério de ouro ao longo de toda a jazida mineral. A escala vertical de cores apresenta o nível de pureza das respectivas regiões que contém o minério de ouro. Esta escala de cores refere-se ao valor em partes por milhão de minério de ouro a qual também pode ser representada como *ppm* de *Au*. Isto quer dizer que se iniciarmos um extração mineral nas regiões que estão representadas pelas cores azul claro até o laranja claro encontraremos, para cada tonelada de minério extraída, um valor que pode variar de uma quantidade de 4,24 a 12,7 gramas de ouro. Vê-se através do modelo de teores do ouro, gerado pelo método da krigagem, apresentado na figura 22 que a sua distribuição ocorre de uma forma bastante uniforme, exceto por duas regiões onde há uma concentração mais significativa do mineral ouro possuindo tonalidades verde e amarelo na região mais centralizada da parte superior da figura e um tom azul claro tendendo para verde no canto superior direito da imagem.

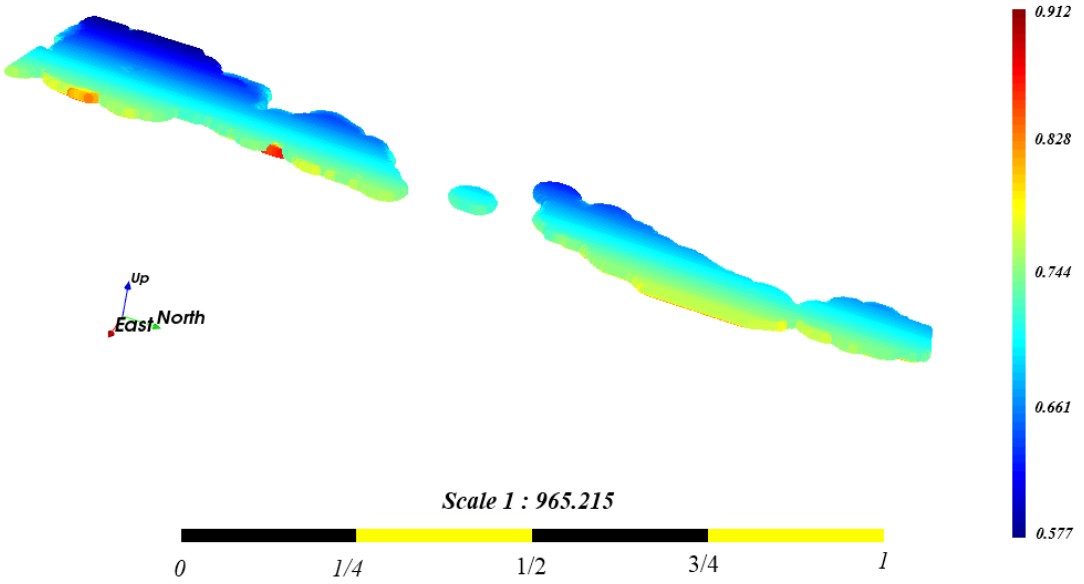

Figura 23 – Modelo de teores de ouro estimado pela regressão linear

A figura 23 representa um modelo geológico de teores para o minério de ouro, sendo esse modelo realizado mediante o treinamento dos dados pelo algoritmo de regressão linear. A escala de cor vertical representa o *ppm* de *Au* ou pureza do minério de ouro encontrado nessa jazida por meio do modelo de regressão linear. Comparando-se a figura 22 com a figura 23 podemos ver que há uma diferença bastante significativa, no que se refere a distribuição espacial das intensidades de teores de ouro ao longo do corpo mineralizado. Isto se deve ao fato de que as hipótese do modelo de regressão linear busca definir a predição de teores por meio de uma função linear. O que não corresponde com a realidade das jazidas minerais, visto que o comportamento da distribuição espacial dos teores de ouro em uma jazida é bastante descontínuo, o que impossibilita ainda mais encontrar um modelo de linear que seja capaz de predizer com precisão a distribuição teores. Isto se confirma através do resultado obtido para o erro da estimativa o qual foi superior ao da krigagem em 13%. Portanto, vê-se que não é possível descrever um fenômeno geológico mineralizado por meio do método da regressão linear, visto que o erro associado a estimativa é significativamente maior que o encontrado para o método tradicional de krigagem.

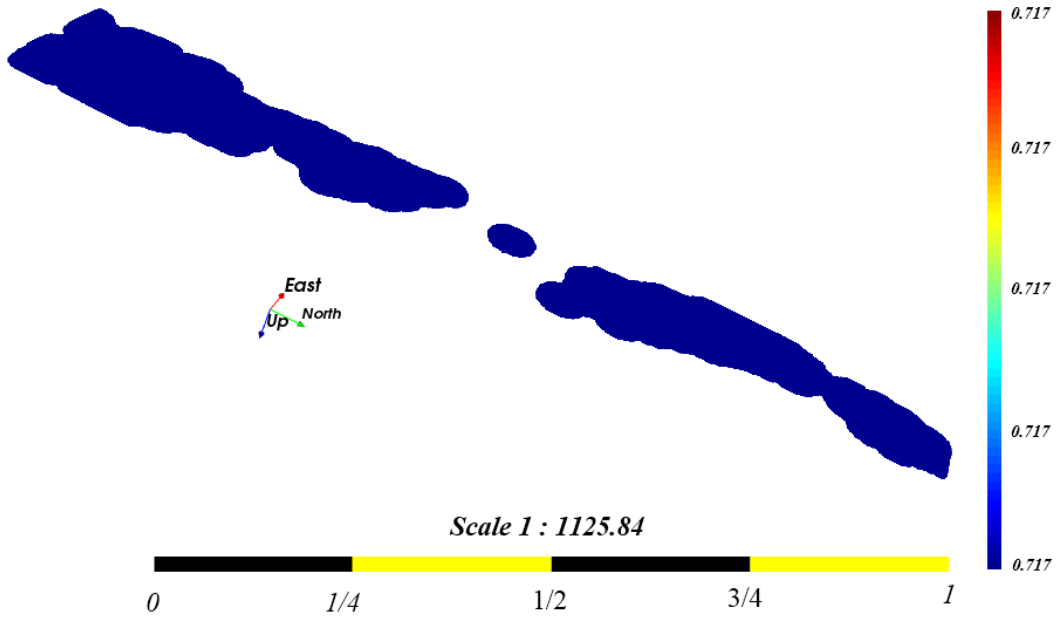

Figura 24 – Modelo de teores de ouro estimado pela regressão lasso

A figura 24 representa o modelo geológico de teores para o minério de ouro. Ao vizualizarmos a imagem, gerada pelo treinamento dos dados de ferro por meio do método de regressão *lasso*, vemos que existe um gradiente constante para a distribuição de teores do corpo mineralizado. Essa constância na representatividade da distribuição de teores é motivada pela restrição algorítmica do modelo *lasso*, a qual pode ser chamada como o fator de regularização, o qual é representado pela restrição do modelo 3.10. Esta restrição faz com que o modelo matemático se torne insenssível a pequenas variações na variável dependente, que neste caso é o teor de minério ouro. Desta forma as variáveis explicativas as quais são representadas pelas coordenadas geográficas X, Y e Z não são relevantes o suficiente para explicar a variação na distribuição espacial do minério de ouro em todo o corpo mineralizado o que se confirma mediante o aumento do erro de estimativa em 12,5%. Essa restrição faz com que a aplicação desse método se torne ainda mais difícil de prever o comportamento espacial de teores ouro, visto que os veios de minério de ouro apresentam bastante descontiuidade o que também pode ser visto na figura acima quando observamos uma região isolada, aproximadamente, no meio do corpo mineralizado.

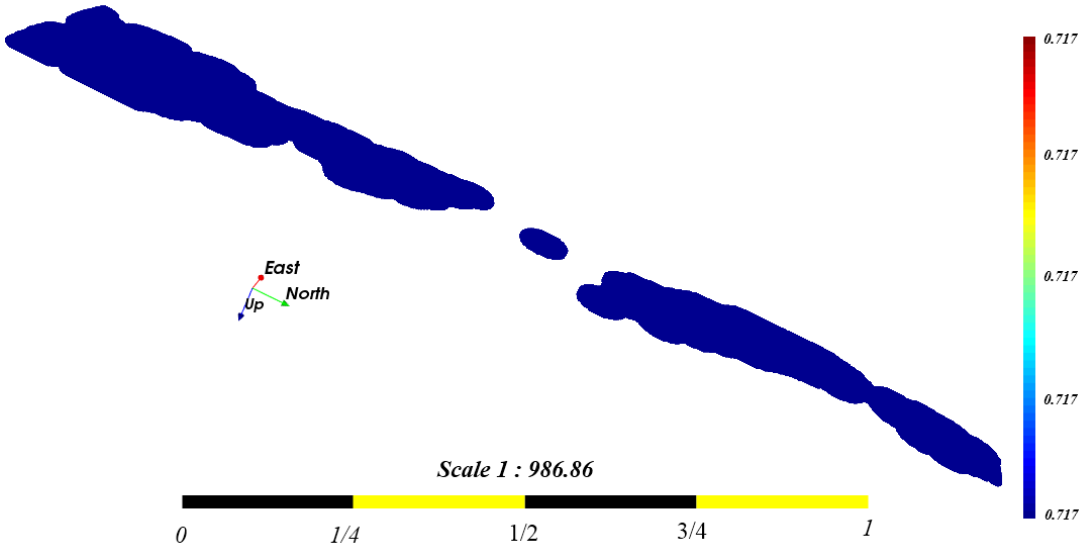

Figura 25 – Modelo de teores de ouro estimado pela regressão EN

A figura 25 mostra o modelo geológico de teores para o minério de ouro com a utilização da metogologia de aprendizado de máquina EN. Observa-ser que o método EN apresenta as mesmas características do $Lasso$, isto é devido ao fator de regulalização dado pela restrição do modelo 3.12. A qual é responsável por diminuir a sensibilidade das variáveis explivativas mediante pequenas alterações da variável explicada, alvo ou dependente. O erro para este método foi superior em 12,5%. Mais uma vez vê-se que os modelos lineares não são se apresentam como bons estimadores de recursos minerais.

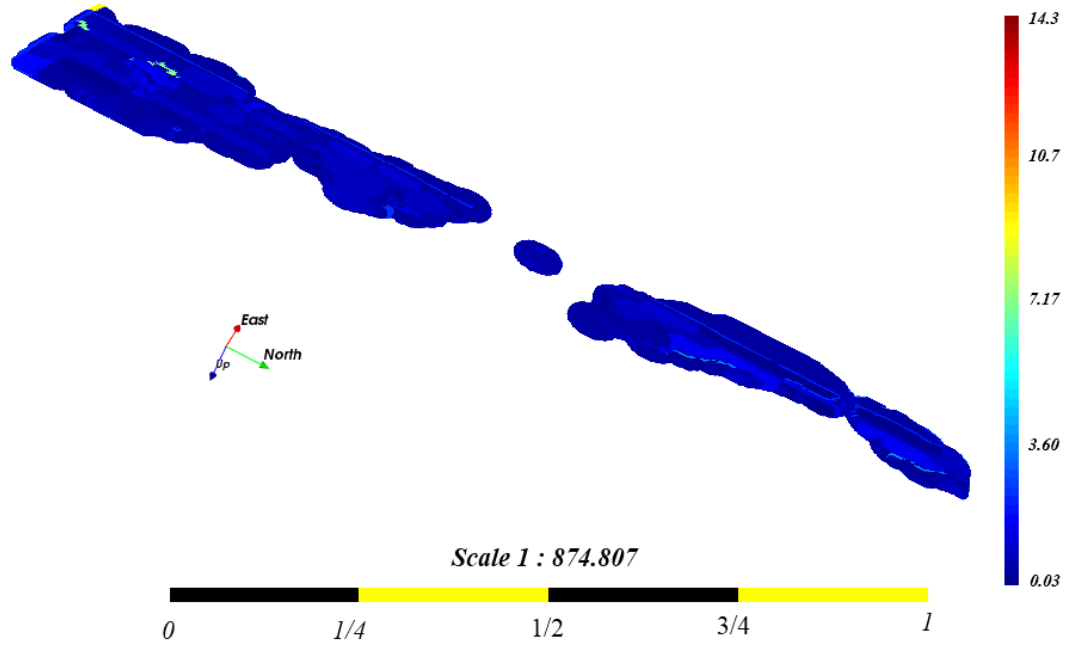

Figura 26 – Modelo de teores de Ouro estimado pelo CART

A figura 26 nos mostra o modelo geológico de teores para o ouro. A escala de cores

apresenta o *ppm* de *Au* ou grau de pureza ditribuída ao longo de todo o corpo mineralizado. Este grau de pureza refer-se ao grau ou nível aurígero encontrado nas diferentes regiões dadas pelas suas respectivas tonalidades de cores. Por exemplo se escolhermos extrair minério da região compreedida entre verde claro e laranja claro encontraremos as quantidades de 7,17 a 10,7 gramas de ouro para cada tonelada extraída do minério respectivamente. Tendo como base treinamento a medida de similaridade, a qual é dada pela equação 3.21, fica claro ao observarmos as imagens 11 e 26 que existem semelhanças significativas para a distribuição de teores de ouro para o corpo mineralizado. Porém, o método $CART$ não se mostrou como um método adequado o suficiente para se realizar estimativas em minério de ouro. Isto é mostrado na tabela 19 onde vemos que o modelo gerou um erro de 5% o qual é ligeiramente superior se compararmos com o obtido com a krigagem. Isto é devido a uma característica particular do algoritmo de árvore de regressão a qual é representada pela baixa capacidade que esse método possui em diminuir a variância do erro das diferentes regiões, isto ocorre porque este modelo de AM trabalha apenas com o treinamento de uma árvore de regressão e por esta razão não possui a capacidade de diminuir a variância do erro para o modelo matemático finalizado. Portanto, mesmo com uma boa capacidade adaptativa e interpretativa para dados espaçados vemos que esse método apresenta um ponto que não contribue para uma estimativa melhorada da jazida de minério de ouro.

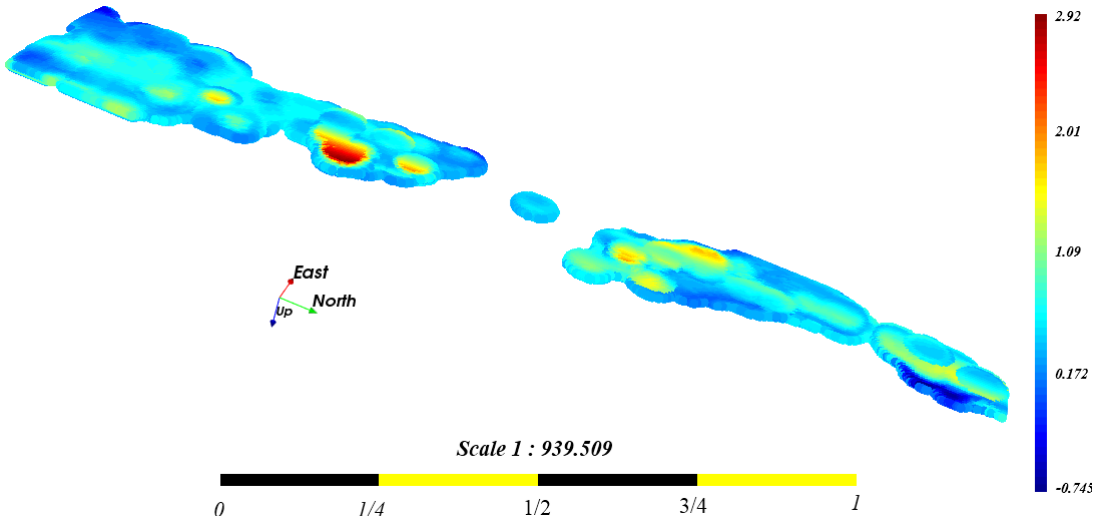

Figura 27 – Modelo de teores de ouro estimado pelo SVR

A figura 27 nos mostra o modelo geológico de teores para o minério de ouro realizado madiante a aplicação do método de aprendizado de máquina SVR. Ao analisarmos as imagens 11 e 27 verificamos que existem várias diferenças de estimativa de teores bastante significativas ao longo grande parte do corpo mineralizado. Pois vemos que grande parte da região mineralizada é representada por uma coloração azul clara tentendo para um verde claro, o que difere bastante se comparada ao modelo gerado pela krigarem. Este

método de AM apesar de ser bastante robusto do ponto de vista de reconhecimento de padrões, fica claro que ele não consegue representar com muita eficácia a distribuição espacial dos teores de ouro ao longo dessa jazida. Isto mais uma vez está associado ao fato desse método possuir um conjunto de funções analíticas ou função kernel K **??** onde essas funções não conseguem descrever o comportamento de natureza aleatória da distribuição do minério de ouro. Entende-se como o padrão da variável de busca, neste contexto, as regiões que possuem características semelhantes no que se refere a variabilidade de distribuição de teor de ouro dentro do corpo mineralizado. Contudo a sua apacidade analítica em avaliar as mínimas diferenças de teores dentro jazida mineral é bastante alta, e isto é um fator importante que o torna um método bastante aceitável para a análise de dados espaçados. O maior problema consiste no conjunto de hipóteses que este método de AM possui, entendamos como hipóteses o conjunto de funções analíticas que o método possui. Portanto, apesar desse método atender ao requisito de suavidade na interpretação dos dados espaçados, fica claro que as suas hipóteses avaliadas dentro desse contexto da estimativa de recursos minerais não contempla uma boa estimativa para a jazida de minério de ferro. Onde pode ser comprovada mediante os resultados mostrados na tabela 19. Vemos que esse método de AM apresentou um diferencial de erro da ordem de 14% a mais do que o obtido com a krigagem, o que se mostra um pouco inferior ao método tradicional.

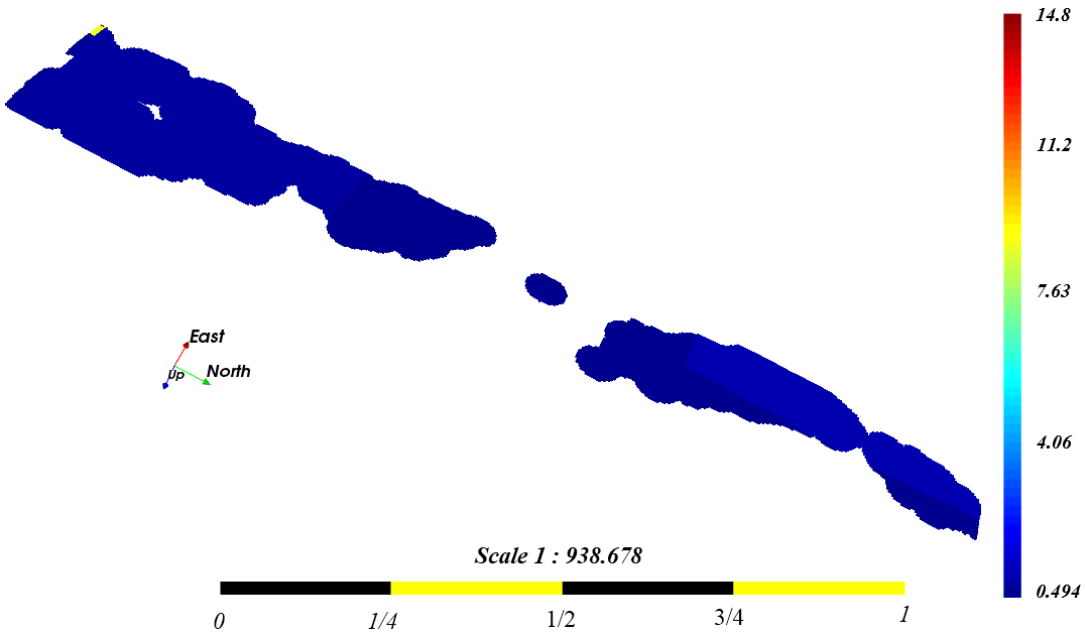

Figura 28 – Modelo de teores de ouro estimado por AdaBoost

A figura 28 apresenta o modelo de estimativa de teores com a utilização do método de AM *AdaBoost*. Ao visualizarmos as imagens 28 e 22 podemos observar que existem diferenças preditórias significativas entre as pequenas regiões que contêm variações mais acentuadas no teor de ouro. O método *AdaBoost* praticamente interpretou toda a região

mineralizada com ouro como sendo um azul tendendo para escuro, o que nos mostra uma invariabilidade na distribuição de teores. Esta invariabilidade de teores ao longo do corpo até certo ponto é significativa, mas ao sobrepormos as imagens vemos que existem duas regiões com uma concentração maior de ouro as quais não foram detectadas esse método de AM. Apesar desse algoritmo trabalhar com um conjunto de regressores e combiná-los de forma a obter uma menor variância para o erro de estimativa, ele apresenta características que o tornam com um viés menos flexível. Essas características referem-se ao conceito da incapacidade de funções analíticas descreverem, com maior precisão, o comportamento da distribuição espacial de teores em uma jazida mineral. Pois, isto acontece com esse método porque ele usa para o treinamento da máquina de aprendizado três funções analíticas que são: *linear*, *quadrática* e *exponencial*. Com isto podemos observar que existe uma semelhança entre o método de krigagem e o método de AM *AdaBoost*, pois os dois se utilizam de funções analíticas para o treinamento dos dados. Porém a principal diferença entre os dois está no quesito de combinação de regressores o que não aconteçe com a krigagem. Vemos, através dos resultados mostrados na tabela 20.

Portanto, vê-se que é possível avaliar recursos minerais por meio do *AdaBoost* e até interpretar de forma satisfatória a distribuição de teores ao longo da jazida, mas ele falha exatamente no quesito de treinamento dos dados espaçados, o que nos mostra de uma forma clara que o comportamento anômalo de uma jazida muitas vezes não dá para ser interpretado por meio de modelos analíticos de funções matemáticas.

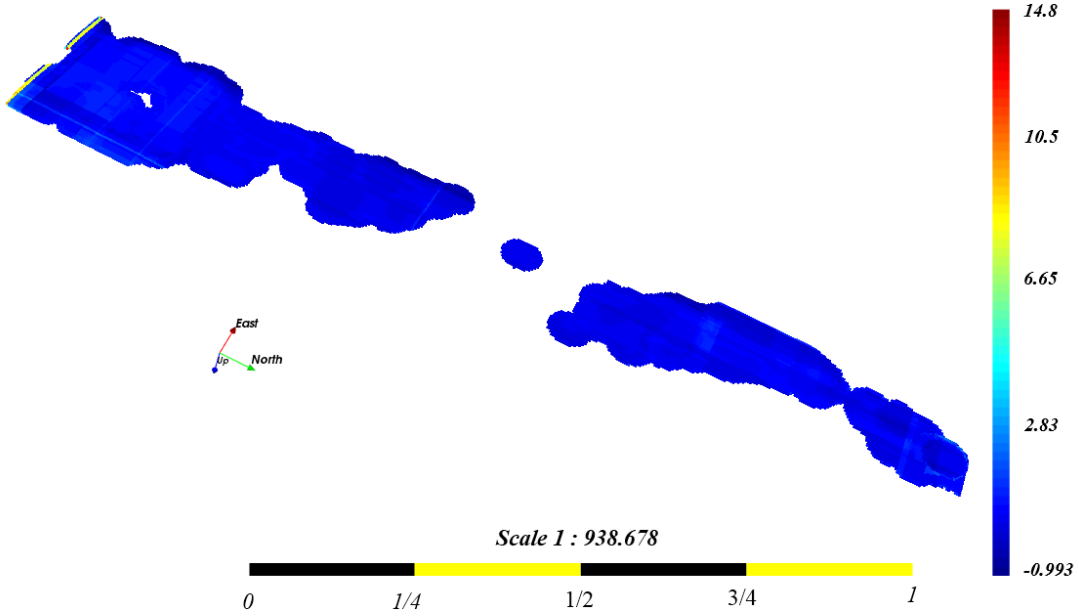

Figura 29 – Modelo de teores de ouro estimado por gradiente boosting

A figura 29 nos mostra modelo de teor gerado por meio da aplicação do algoritmo de AM Gradient Boosting. Se contrastarmos as figuras 29 com 22 veremos que existe uma predominância da cor azul escura ao longo do corpo mineralizado. Isto se deve a forma de

treinamento desse algoritmo visto que ele utiliza o método de gradiente descendente para conseguir encontrar os modelos otimizados, ou seja, encontrar o conjunto de regressores que melhor definem o comportamento do fenômeno estudado, que neste caso específico representa a distribuição de teores ao longo de uma jazida de minério de ouro. Este método, mesmo tendo características de treinamento parecidas com o *AdaBoost*, se mostra mais eficiente para a avaliação de dados espaçados, como mostra os resultados da tabela 20 da qual podemos extrair que esse método teve um diferencial de erro da ordem de 10% a mais que o método treadicional. De toda forma a metodologia de treinamento desse algoritmo não se mostra adequada para o estudo de estimativa de recursos minerais, visto que ela apresenta um problema que está na dificuldade de o gradiente descendente encontrar o ótimo global para cada uma das hipótese ou regressores. Portanto, ainda que seja um método viável para a análise de recursos minerais ele falha nesse quesito do gradiente descendente.

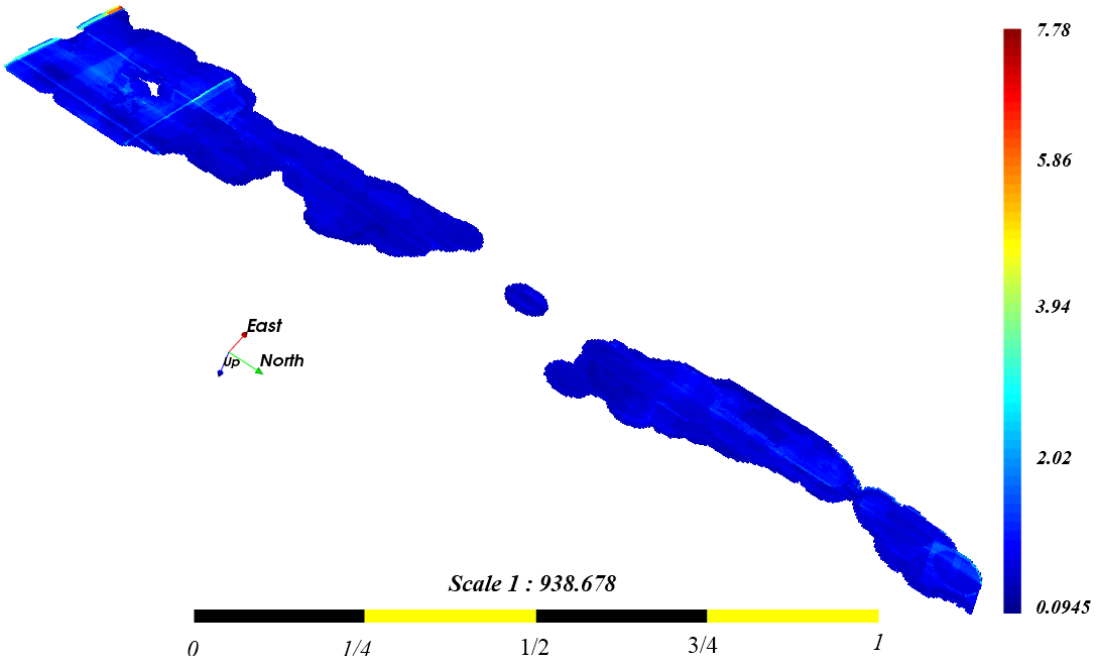

Figura 30 – Modelo de teores de ouro estimado por florestas aleatórias

A figura 30 mostra o modelo geológico de teores para o minério de ouro realizado com a utilização do método de AM florestas aleatórias o qual também é conhecido como *random forest*. Ao compararmos as figuras 22 com a figura 30 verificamos que existem algumas diferenças no que concerne a distribuição espacial de teores no corpo mineralizado. Porém, observamos também que existem semelhanças significativas nas regiões onde contém um grau mais alto do teor de ouro, as quais se localizam na parte superior esquerda da imagem e também um pouco acima da região central superior esquerda da figura. Mediante os resultados obtidos vemos que o algoritmo de florestas aleatórias conseguiu ser melhor para a estimativa da distribuição de teores de ouro ao longo do corpo de minério. Uma

vantagem bastante significativa que o método floresta aleatórias apresenta sobre o método da krigagem é o fato dele não fazer nenhuma hipótese acerca da distribuição espacial dos dados, ou seja, ele não condiciona o treinamento a nenhuma função analítica, o que se mostra como um método muito bom para a análise de estimativa de recursos minerais e isto é muito importante para a análise em jazidas auríferas, visto que esse tipo de jazida apresenta bastante descontinuidade na distribuição do teor no corpo mineralizado. O sucesso do método florestas aleatórias se dá exatamente porque ele consegue combinar os diversos regressores, que neste caso é um conjunto de árvores de regressão, com a sua capacidade de otimizar as partições de conjuntos de dados para cada *nó* mediante a aquação 3.21. Isto é refletido na diminuição do erro de estimativa da ordem de 16%.

Portanto, esse método nos fornece um novo olhar ou olhar melhorado no campo de análise de recursos minerais visto que ele se aprensenta como uma técnica bastante robusta para o estudo de dados espaçados.

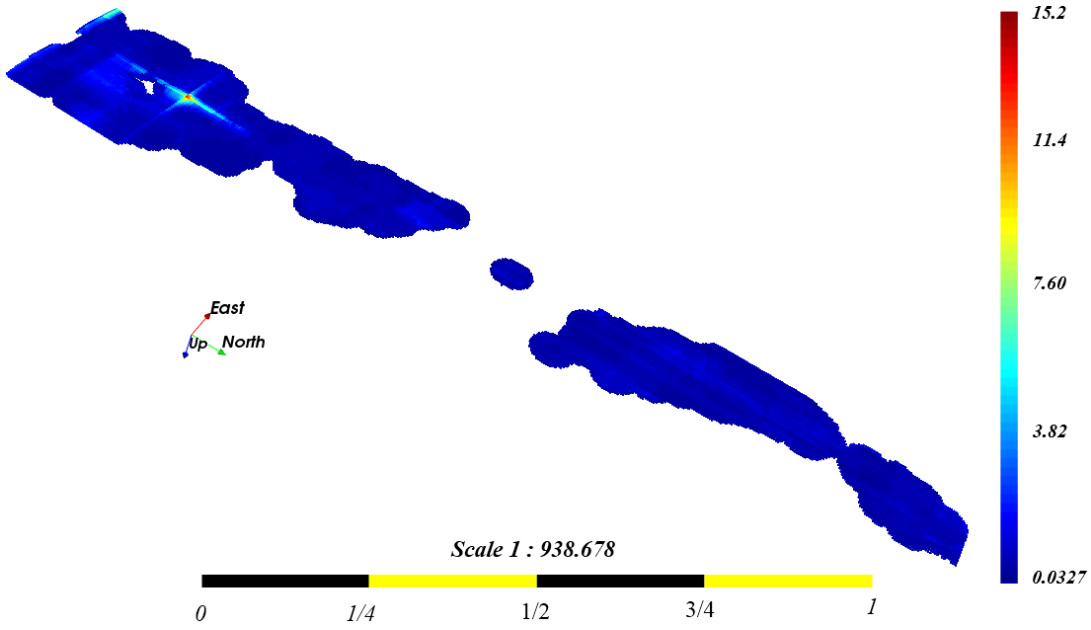

Figura 31 – Modelo de teores de ouro estimado por florestas extra aleatórias

A figura 31 mostra o modelo geológico de teores para o minério de ouro realizado com a aplicação do método de AM florestas extra aleatórias, o qual é chamado de *Extra Trees*. Fazendo-se uma análise comparativa das imagens 22 e 31 observamos algumas variações para a distribuição de teores ao longo do corpo mineralizado. Contudo, se contrapusermos as imagens geradas pelos métodos de krigagem, florestas aleatórias e florestas extra aleatórias 22, 30 e 31 respectivamente, observaremos que existe o mesmo ponto comum que é a semelhança de localização de região que contém o maior teor de minério. Isto se confirma pela propriedade de minimização da variância do erro que ocorre nos três métodos supracitados.

O método florestas extra aleatórias se mostrou mais eficiente e eficaz para a estimativa do minério de ouro nessa jazida. Isto se deve a presença de uma uma característica que o torna um método ainda melhor que o de florestas aleatórias, essa característica é a sua forma de treinamanto dos dados, pois esta metodologia de AM trata cada conjunto de dados para treinamento de uma forma completamemte aleatória o que fornece uma flexibilidade ainda mais alta para a máquina de aprendizado finalizada ou modelo de estimativa final. Esta propriedade faz com que esse método seja capaz de trabalhar com jazidas com uma alta descontinuidade ou erráticas. Portanto este método é perfeitamente aplicável para a análise de recursos minerais sejam eles contínuos ou descontínuos. Verificamos através da tabela 20 que ele gerou um deferencial de erro para a estimativa da ordem de 22,5% inferior ao método tradicional, o que se confirma como uma metodologia perfeitamente aplicável em estimativas de recursos minerais.

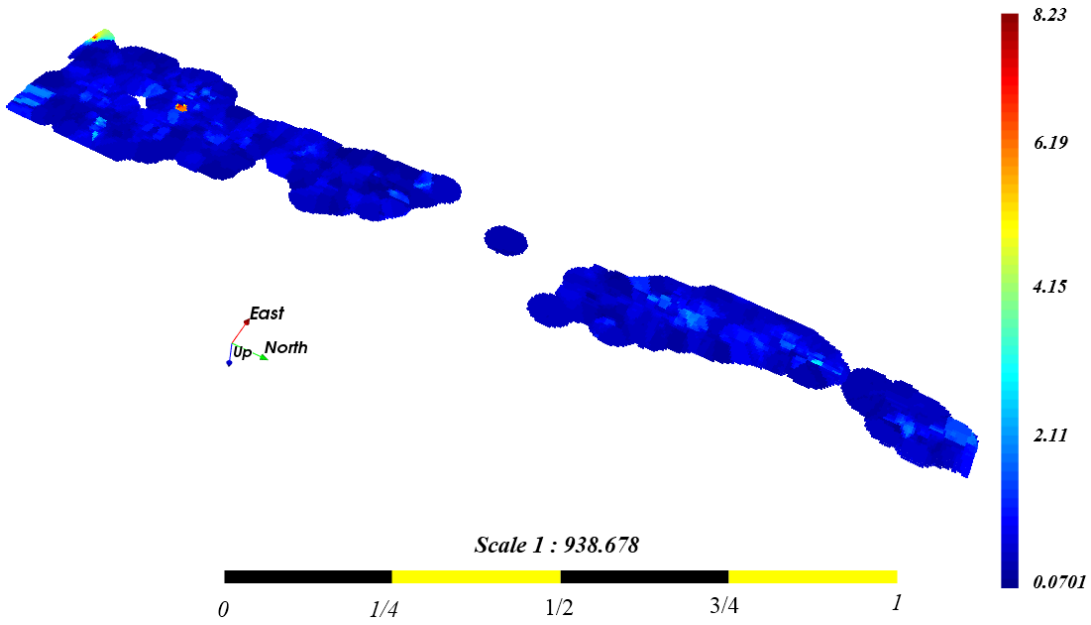

Figura 32 – Modelo de teores de ouro estimado por vizinho mais próximo

A figura 32 mostra o modelo geológico de teores para o minério de ouro obtido por meio do método de AM $K - NN$ o qual é mais conhecido como os K vizinhos mais próximos. O $K - NN$ é um método relativamente simples onde cada ponto é estimado em termos dos teores dos pontos mais próximos ou vizinhos mais próximos. Conparando-se a imagen 22 com a imagem 32 visualizamos que há regiões semelhantes ao longo do corpo mineralizado. O fato de um algoritmo relativamente simples apresentar um erro de estimativa menor que o encontrado no método tradicional de krigagem reside no fato de utilizarmos vários algoritmos para o cálculo do vizinho mais próximo, pois no treinamento dessa máquina de aprendizado vemos que o algoritmo *auto* se mostrou melhor para mapear os dados de ferro e níquel ao passo que o *ball tree* foi mais bem aplicado para o ouro, como mostrado na tabela 9. Portanto, observa-se que existe uma forte correlação espacial entre

os teores de minério de ouro. Em se tratando de jazidas com descontinuidade acentuada essa correlação espacial não é apresentada ao longo de todo o corpo mineralizado, mesmo assim temos que essa algoritmo se apresentou mais eficaz para a medição do teor de ouro ao longo de toda a região mineralizada. Ao avaliarmos a tabela 19 vemos que houve uma diminuição do erro de estimativa de minério de ouro da ordem de 19,6%. Portanto, esse método de AM serve como um referencial para avaliar características semelhantes em outros métodos de AM para aplicação em estimativa de recursos minerais auríferos.

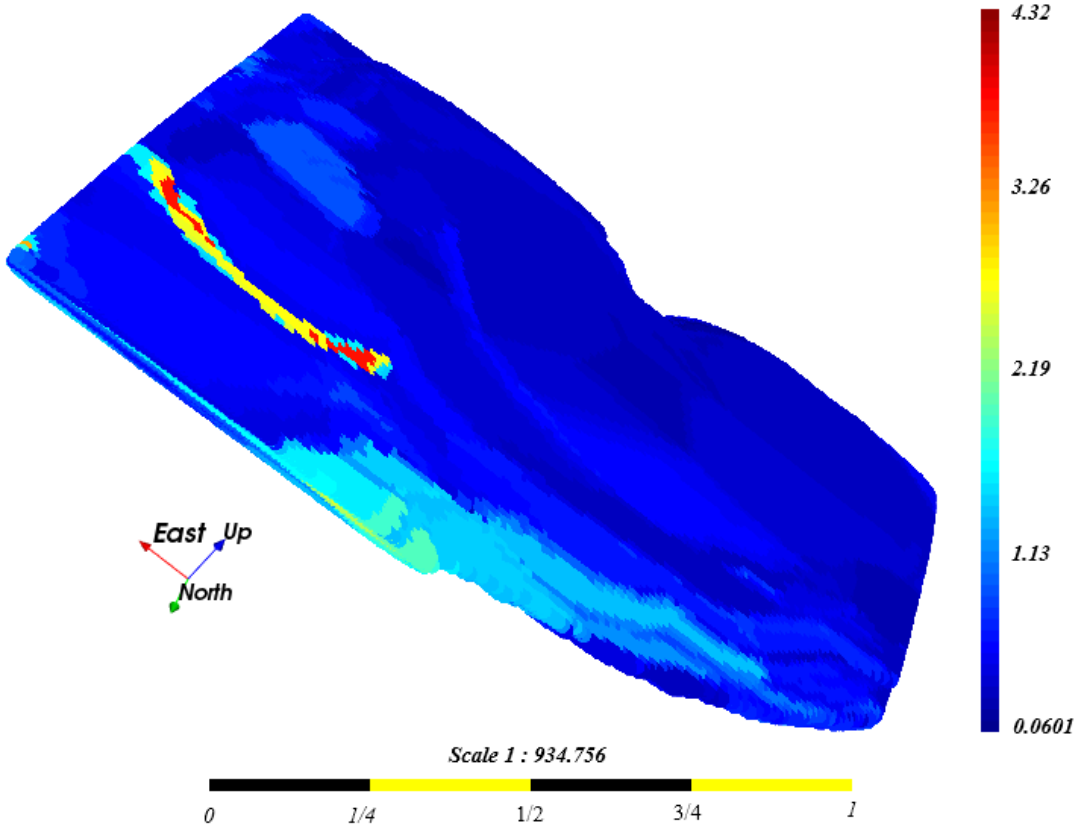

Figura 33 – Modelo de teores de níquel estimado pela krigagem

O modelo geológico de teores apresentado acima mostra a distrtibuição espacial do minério de níquel ao longo de toda a jazida mineral. A escala vertical de cores apresenta o nível de pureza das respectivas regiões que contém o minério de níquel. Por exemplo, se iniciarmos um extração mineral entre as regiões que estão compreendidas pelas cores azul claro a laranja encontraremos, para cada tonelada de minério extraída, um valor que pode variar de uma quantidade de 11,3 a 32,6 kilogramas de minério de níquel será lavrado ou extraído. O que corresponde, na escala de cores, a um percentual de 1,13% a 3,26% respectivamente. Vê-se através do modelo de teores apresentado acima que a distribuição dos mesmos ocorre de forma bastante contínua dentro corpo mineralizado. Exceto pela ocorrência de uma concentração maior de minério de níquel na parte superior esquerda da figura 33 onde é mostrado um teor mais elevado do níquel dentro da jazida.

Os resultados obtidos com a aplicação do método de krigagem ordinária, tabela 18, nos mostra que essa metodologia de estimativa de recursos minerais se mostrou inferior, em termos de erro de estimativa, dos métodos de KNN, florestas aleatórias e florestas extra aleatórias. Isto se deve a capacidade que o algoritmo de krigagem ordinária apresenta em diminuir a váriância em seu método de interpolação, o que naturalmente reflete bons resultados em análise de dados espaçados. Porem, a concepção de hipóteses muito fortes acerca do comportamento da distribuição espacial dos teores em subsolo compromete a qualidade do modelo de teores gerado. Pois, há uma certa imprevisibilidade na adaptação das jazidas minerais às hipópteses variográficas existentes na literatura. Portanto, vemos que com esse método podemos realizar, com um grau de certeza relativamente bom, estimativas em dados espaçados, que é caso de dados provenientes de sondagens de jazidas minerais. Isto se deve ao fato desse método impor a condição de variância mínima. Mas pelo outro lado ele nos obriga a utilizar as suas funções variográficas preestabelecidas, o que às vezes impossibilita a realização da estimativa devido ao comportamento errático ou com muita descontinuidade da jázida mineral. No caso do níquel vê-se que há uma continuidade bastante forte o que nos mostra que ele é um método adequado para jazidas minerais que possuem esse comportamento.

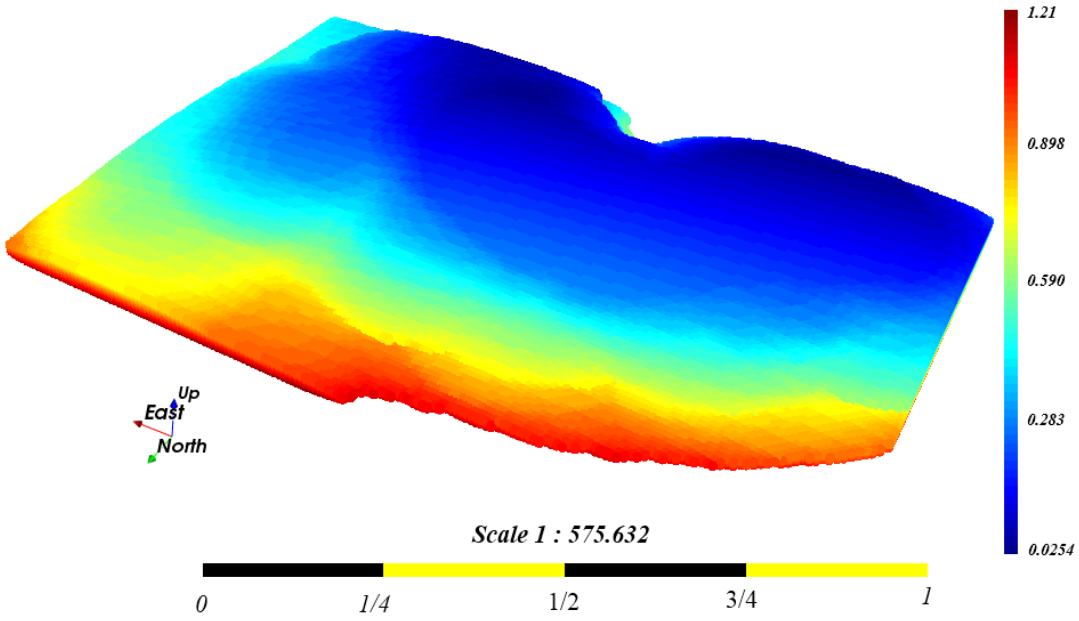

Figura 34 – Modelo de teores de níquel estimado por regressão linear

A figura 34 representa um modelo geológico de terores para o minério de níquel, sendo esse modelo realizado mediante o treinamento dos dados pelo algoritmo de regressão linear. A escala de cor vertical representa o percentual de teor ou pureza do minério de níquel encontrado nesta jazida por meio desse modelo de regressão linear. Comparando-se a figura 34 com a figura 33 podemos ver que há uma diferença bastante significativa, no que se refere a distribuição espacial das intensidades de teores. Isto reflete o viés muito

forte que modelo de regressão linear apresenta, ou seja, este modelo tenta interpretar toda a distribuição espacial de teores de níquel na jazida por meio de um modelos lineares. O que não corresponde com a natureza das jazidas minerais. Portanto, vê-se que não é possível descrever um fenômeno geológico mineralizado por meio do método da regressão linear, visto que o erro associado as estimativas, tabela 18, possue um diferencial superior da ordem de 90%.

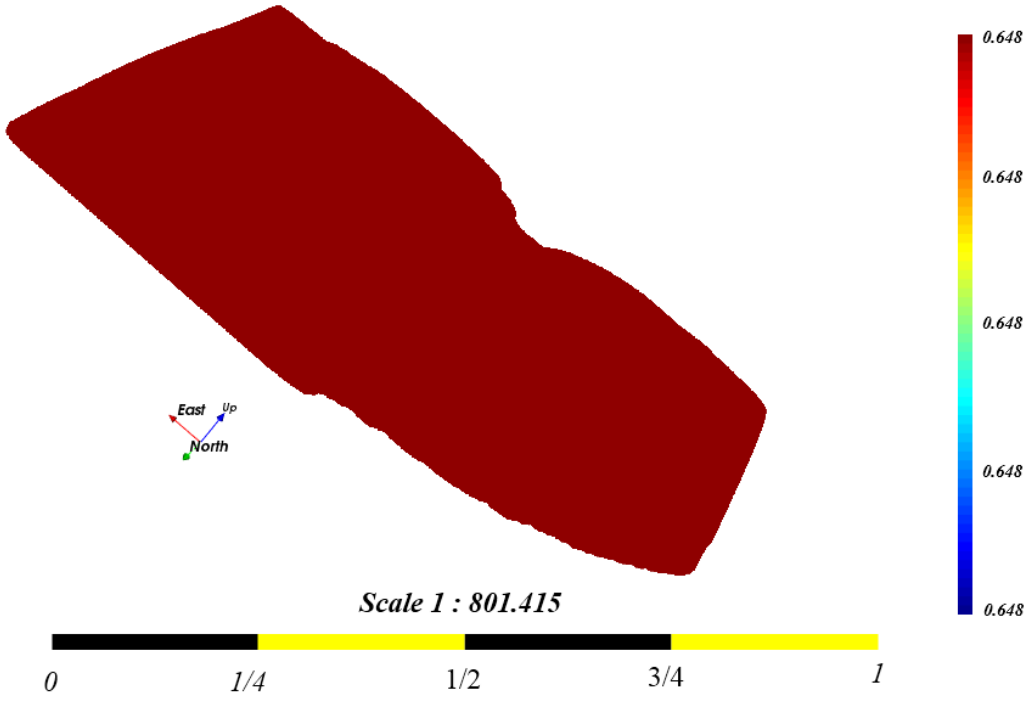

Figura 35 – Modelo de teores de níquel estimado pela regressão lasso

A figura 35 representa o modelo geológico de teores para o minério de níquel. Podemos vizualizar por meio dessa imagem que o modelo matemático gerado pelo treinamento dos dados de ferro através do método de regressão *lasso* apresenta novamente um gradiente constante para a distribuição de teores do corpo mineralizado. Essa constância na representatividade da distribuição de teores é motivada pela restrição que o modelo matamático de treinamento *lasso* possui. Essa restrição pode ser chamada como o fator de regularização, o qual é representado pela restrição do modelo 3.10. Esta restrição torna o modelo matemático muito insenssível a pequenas variações na variável dependente, que para este caso é o teor de minério níquel. Desta forma as variáveis explicativas as quais são representadas pelas coordenadas geográficas X, Y e Z não são relevantes o suficiente para explicar a variação na distribuição espacial de minério de níquel em todo o corpo mineralizado. A tabela nos mostra que o método *lasso* apresentou um diferencial superior de erro da ordem de 117%.

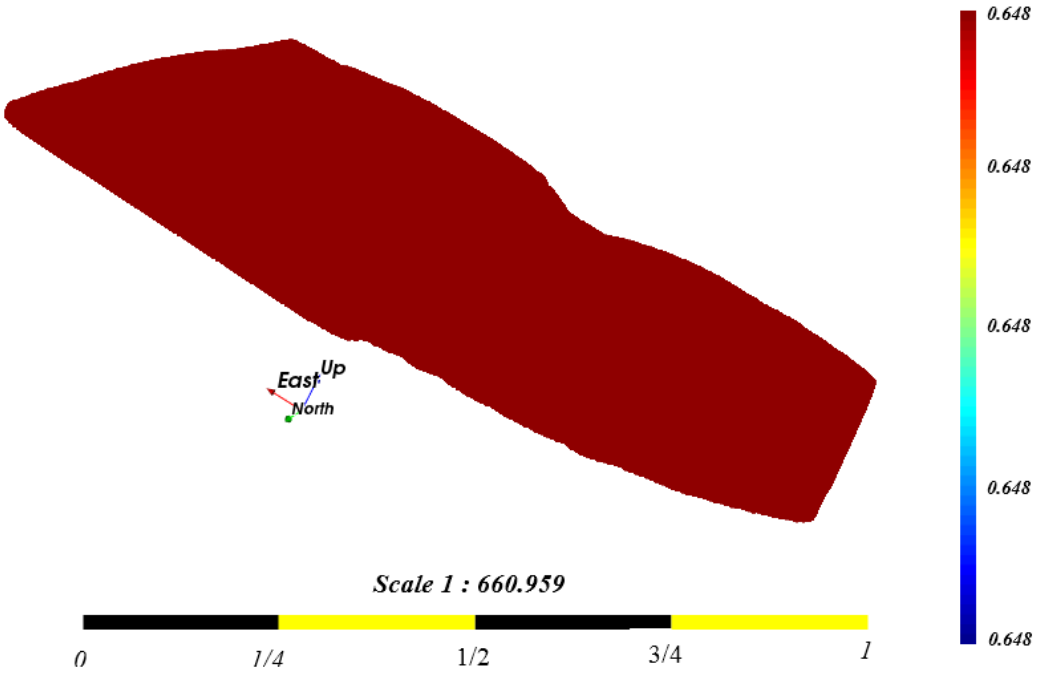

Figura 36 – Modelo de teores de níquel estimado pela regressão EN

A figura 36 mostra o modelo geológico de teores para o minério de níquel com a utilização da metogologia de aprendizado de máquina EN. Podemos observar que o método EN apresenta as mesmas características do $Lasso$, isto é devido ao fator de regulalização dado pela restrição do modelo 3.12. A qual é responsável por diminuir a sensibilidade das variáveis explivativas mediante pequenas alterações da variável explicada, alvo ou dependente. Portanto, vemos que os modelos lineares não são bons representantes para a elaboração de modelos geológico de teores para o minério de ferro.

O EN produziu um erro da ordem de 117% mais alto que o método tradicional da krigagem. É observado que os métodos de AM lineares não apresentam boa performance para estimativas de recursos minerais visto que esses métodos apresentaram erros bem mais elevados do que o método de krigagem.

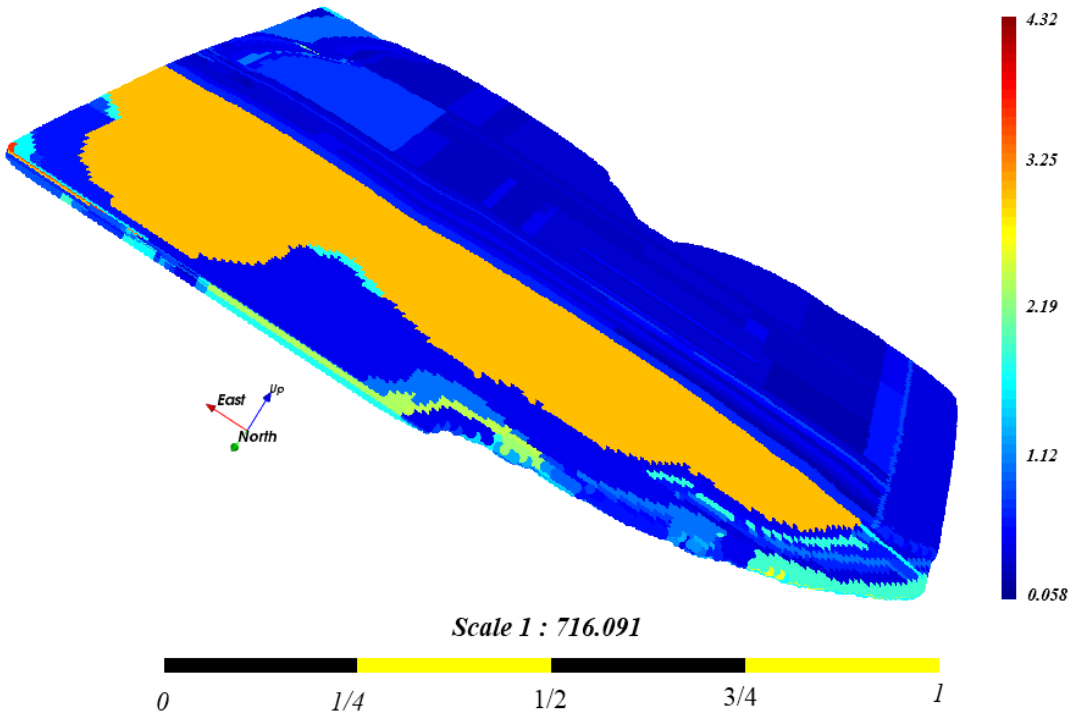

Figura 37 – Modelo de teores de níquel estimado pelo CART

A figura 37 nos mostra o modelo geológico de teores para o níquel. Neste modelo a escala de cores apresenta o percentual de pureza ditribuída ao longo de todo o corpo mineralizado. Este grau de pureza refer-se ao grau ou nível de níquel encontrado nas diferentes regiões dadas pelas suas respectivas tonalidades de cores. Ao vizualizarmos as imagens 33 e 37 verificamos que existem diferenças bastante significativas entre as regiões, onde a figura 37 apresenta variações muito intensas de uma região para a outra se comparada com figura 33. Isto é devido a uma característica particular do algoritmo de árvore de regressão a qual é: a baixa capacidade que esse método possui em diminuir a variância do erro das diferentes regiões, o que é confirmado mediante as comparações entre as imagens supracitadas. Isto ocorre porque este modelo de AM trabalha apenas com o treinamento de uma árvore de regressão e por esta razão não possui baixa capacidade de diminuir a variância do erro para o modelo matemático finalizado. Portanto, mesmo com uma boa capacidade adaptativa e interpretativa para dados espaçados vemos que esse método apresenta um ponto que não contribue para uma estimativa melhorada da jazida de minério de níquel e isto é mostrado mediante o resultado que se encontra na tabela 19, o qual foi superior em 25% se comparado ao método de krigagem.

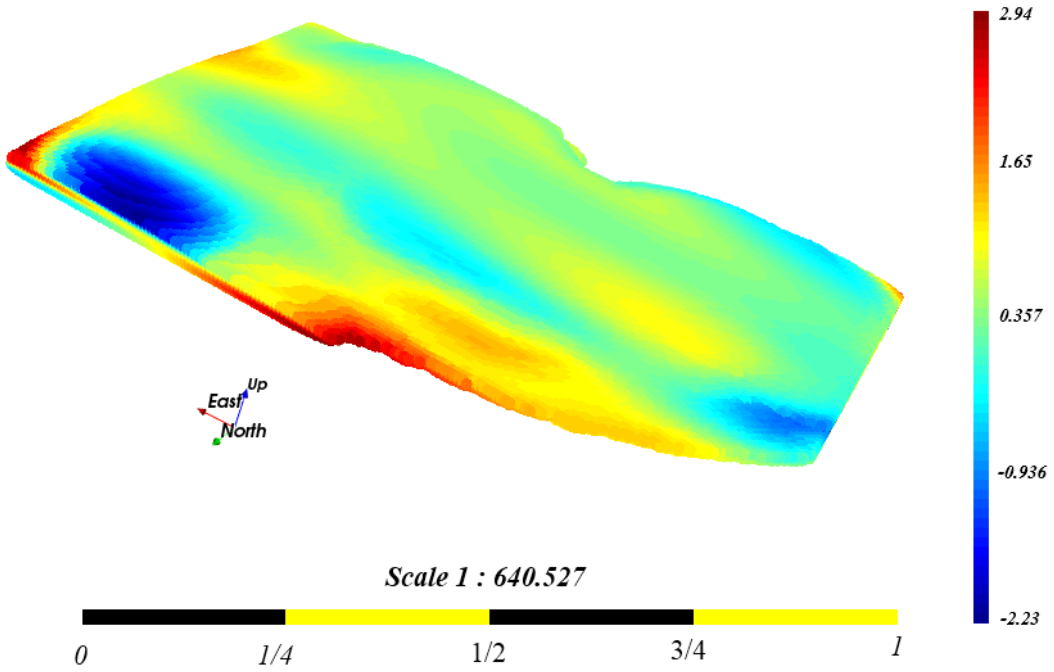

Figura 38 – Modelo de teores de níquel estimado pelo SVR

A figura 38 nos mostra o modelo geológico de teores para o minério de níquel realizado madiante a aplicação do método de aprendizado de máquina SVR. Ao observarmos as imagens 33 e 38 verificamos que existem diferenças de estimativa de teores bastante significativas ao longo grande parte do corpo mineralizado. Contudo a sua apacidade analítica em avaliar as mínimas diferenças de teores dentro jazida mineral é bastante alta, isto é mostrado pela suavidade com ocorre as variações de cores ao longo da jazida. Isso é um fator importante que o torna um método bastante robusto para a análise de dados espaçados. O maior problema consiste no conjunto de hipóteses que este método de AM possui, entendamos como hipóteses o conjunto de funções analíticas que o método possui. Portanto, apesar desse método atender ao requisito de suavidade na interpretação dos dados espaçados para o minério de níquel, fica claro que as suas hipóteses avaliadas dentro desse contexto da estimativa de recursos minerais não contempla uma boa estimativa para a jazida de minério de ferro. Onde esta afirmativa pode ser comprovada mediante os resultados mostrados na tabela 19, onde o erro foi da ordem de 83% a mais do que o obtido com a krigagem.

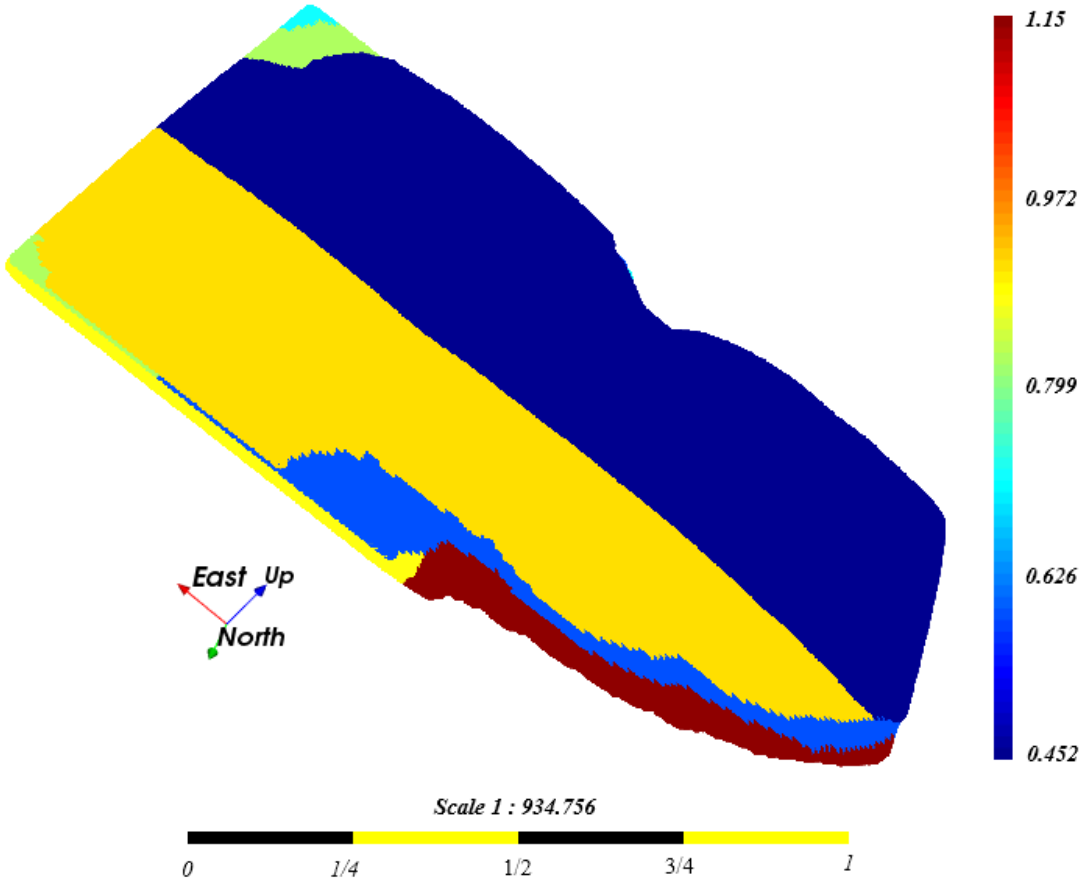

Figura 39 – Modelo de teores de níquel estimado por AdaBoosting

A figura 39 apresenta o modelo de estimativa de teores com a utilização do método de AM *AdaBoost*. Ao visualizarmos as imagens 33 e 39 podemos observar que existem diferenças preditórias, ou seja, de estimativas entre as regiões do corpo mineralizado. Apesar desse algoritmo trabalhar com um conjunto de regressores e combiná-los de forma a obter uma menor variância para o erro de extimativa, ele apresenta características que o tornam com um viés menos flexível, e isto é mostrado através do modelo de teores níquel gerado por ele. Isso acontece com esse método porque ele usa para o treinamento da máquina de aprendizado três funções analíticas que são: *linear*, *quadrática* e *exponencial*. Com isto podemos observar que existe uma semelhança entre o método de krigagem o método de AM *AdaBoost* porém a principal diferença entre os dois está no quesito de combinação de regressores o que não aconteçe com a krigagem. Vemos, através dos resultados mostrados na tabela 20 que existe um erro 87% mais alto se comparado ao da krigagem. Esse método de AM não gerou bons resultados para a análise dos dados de níquel visto que ele apresentou um erro bastante alto para esta jazida. O que nos mostra de uma forma clara que o comportamento anômalo de uma jazida muitas vezes não dá para ser interpretado por meio de modelos analíticos de funções matemáticas.

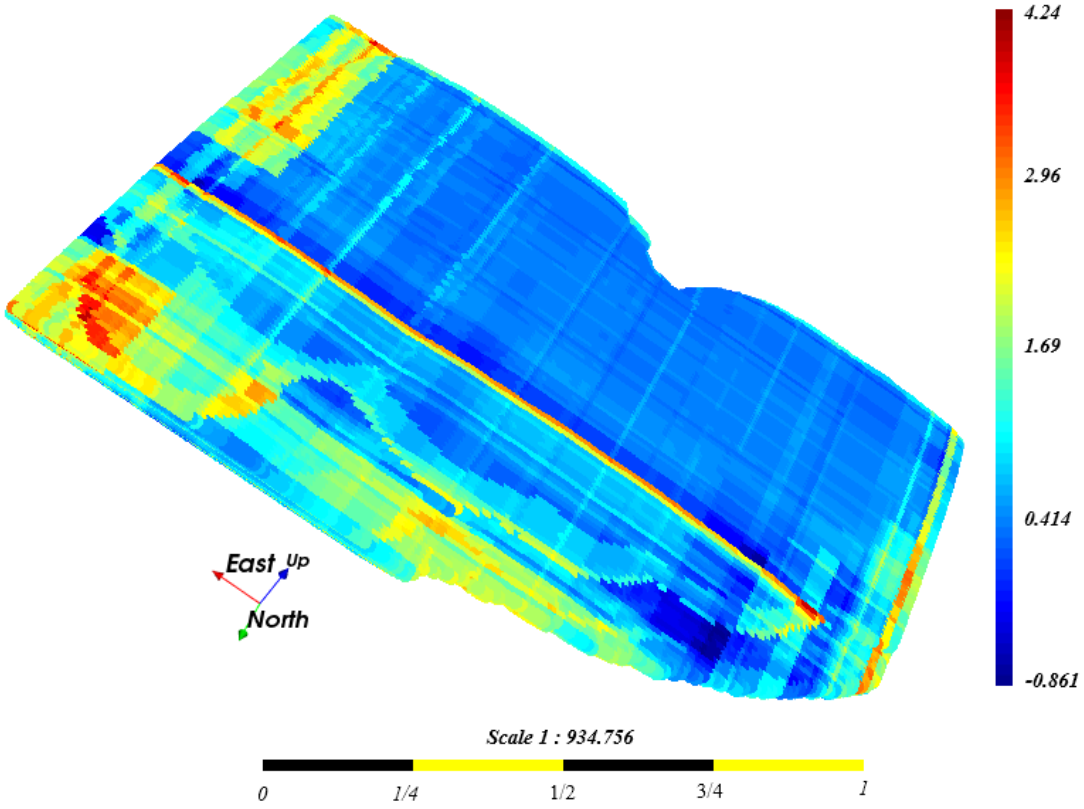

Figura 40 – Modelo de teores de níquel estimado por gradiente boosting

A figura 40 nos mostra modelo de teores do níquel gerado por meio da aplicação do algoritmo de AM Gradient Boosting. Se contrastarmos as figuras 40 com 33 veremos que existem diferenças nas configurações de cores ao longo do corpo mineralizado. Este método, mesmo tendo características de treinamento parecidas com o *AdaBoost*, se mostra mais eficiente para a avaliação de dados espaçados. Isto se deve a forma de treinamento desse algoritmo visto que ele utiliza o método de gradiente descendente para conseguir encontrar os modelos otimizados, ou seja, encontrar o conjunto de regressores que melhor definem o comportamento do fenômeno estudado, que neste caso específico representa a distribuição de teores de níquel ao longo de uma jazida de minério de níquel. Mas esta forma de treinamento ainda que se mostra adequada para o estudo de estimativa de recursos minerais, visto que ela obteve um erro relativamente baixo se comparado com o método tradicional, este erro foi da ordem de 38% maior que o da krigagem. Portanto, ainda que seja um método viável para a análise de recursos minerais ele falha nesse quesito do gradiente descendente.

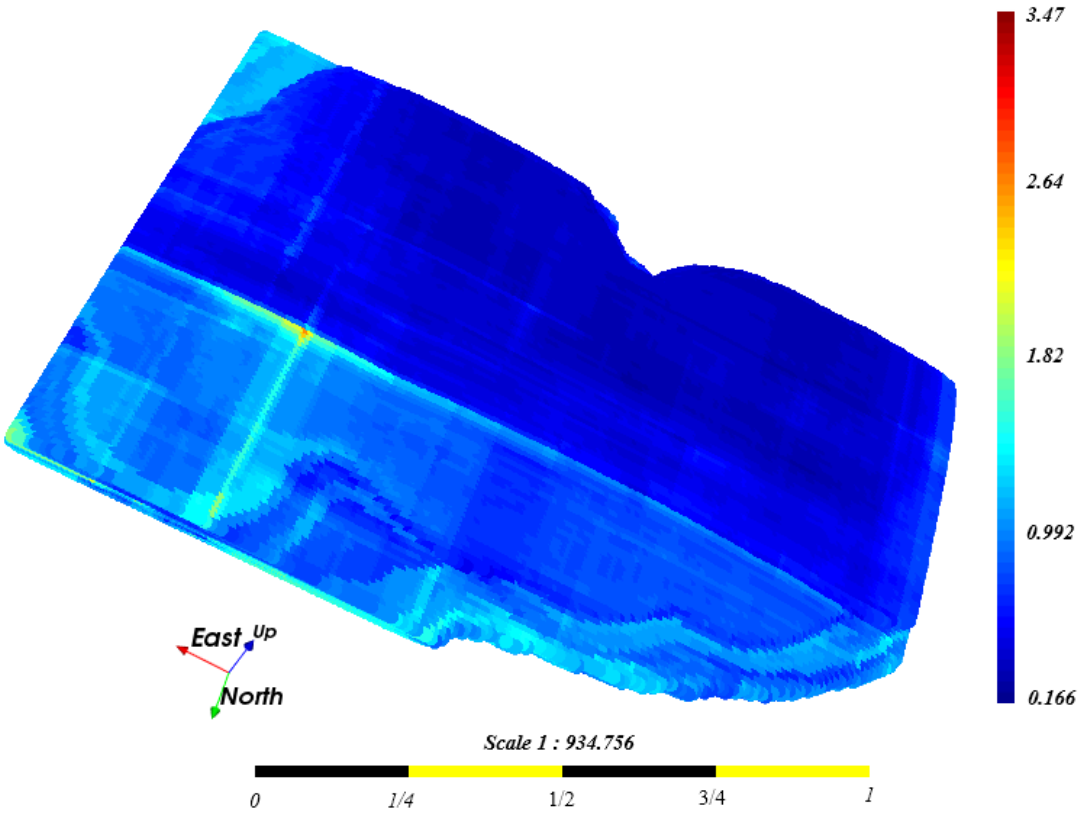

Figura 41 – Modelo de teores de níquel estimado por florestas aleatórias

A figura 41 mostra o modelo geológico de teores para o minério de níquel realizado com a utilização do método de AM florestas aleatórias ou simplesmente chamado de *random forest*. Ao compararmos as figuras 33 com a figura 41 verificamos que existem algumas diferenças entre as distribuições espaciais de teores no corpo mineralizado. Observamos também que há poucas semelhanças no que se refere a localização máxima de teores de níquel. Pois, o modelo gerado pelo algoritmo de florestas aleatórias apresentou uma menor concentração de minério de níquel onde o método tradicional apresentou um filão com maior intensidade. Este modelo conseguiu diminuir o erro de estimativa em 15,6% o que representa uma vantagem bastante significativa sobre o método da krigagem. Isto se deve ao fato do não condicionamento de hipóteses para a realização do treinamento dos dados. O que se mostra novamente como um método muito bom para a análise de estimativa de recursos minerais. O sucesso do método florestas aleatórias se dá exatamente porque ele consegue combinar os diversos regressores, que neste caso é um conjunto de árvores de regressão, com a sua capacidade de otimizar as partições de conjuntos de dados para cada *nó* mediante a aquação 3.21. Portanto, esse método nos fornece um olhar melhorado no campo de análise de recursos minerais visto que ele se aprensenta como uma técnica bastante robusta para o estudo de dados espaçados.

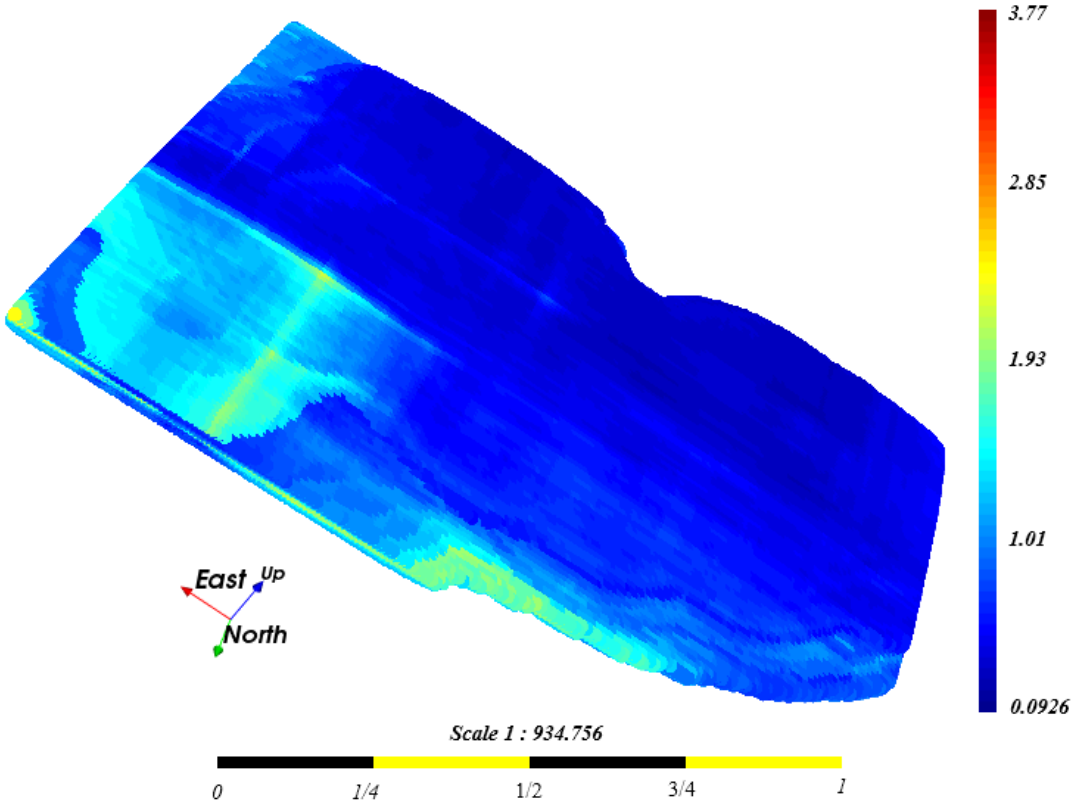

Figura 42 – Modelo de teores de níquel estimado por florestas extra aleatórias

A figura 42 mostra o modelo geológico de teores para o minério de níquel realizado com a aplicação do método de AM florestas extra aleatórias, o qual é simplesmente chamado de *Extra Trees*. Fazendo-se uma análise comparativa das imagens 33 e 42 observamos variações significativas para a distribuição de teores ao longo do corpo mineralizado. Contudo, se contrapusermos as imagens geradas pelos métodos de krigagem, florestas aleatórias e florestas extra aleatórias 33, 41 e 42 respectivamente, observaremos que existe mais semelhanças ou regiões parecidas entre os métodos de florestas aleatórias e extra aleatórias. Apesar da propriedade de minimização da variância do erro que ocorrer nos três métodos supracitados vemos através dos resultados na tabela 20 que o algoritmo florestas extra aleatórias foi superior em relação ao método de krigagem e florestas aleatórias, onde obteve um percentual de diminuição do erro da ordem de 24% e 15,6 % respectivamente.

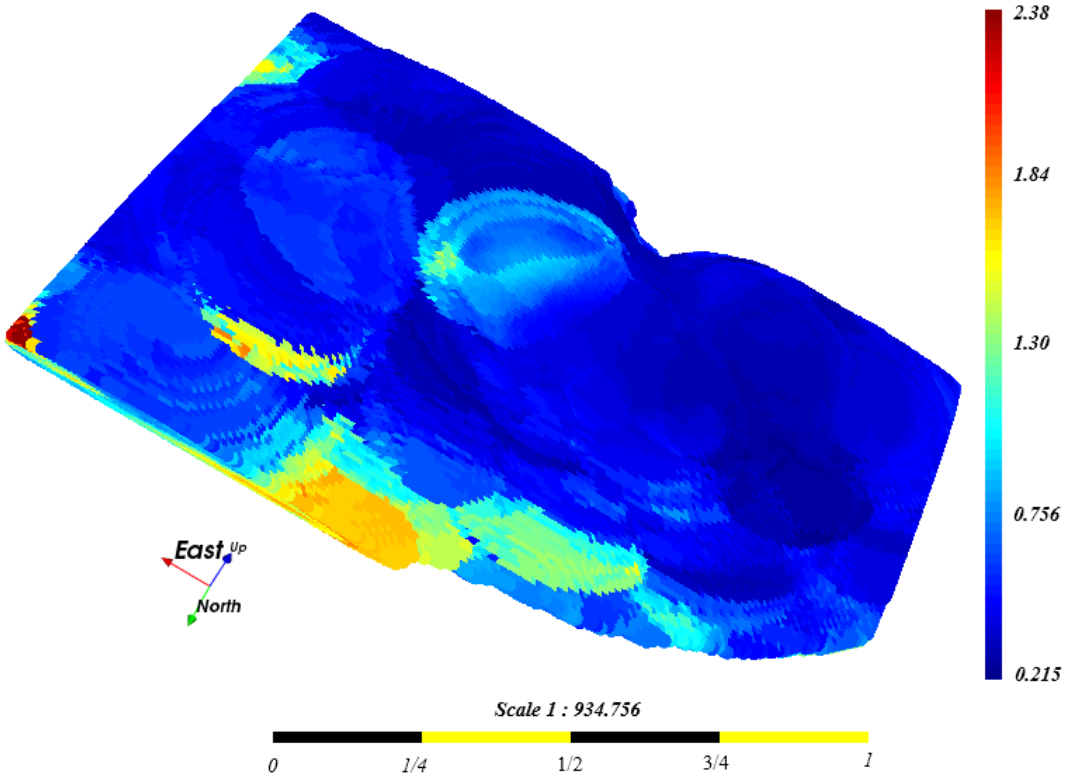

Figura 43 – modelo de teores de níquel estimado por vizinho mais próximo

A figura 43 mostra o modelo geológico de teores para o minério de níquel obtido mediente a aplicação do método de AM $K - NN$ o qual é mais conhecido como os K vizinhos mais próximos, onde K representa o número de pontos mais próximos do ponto que desejamos estimar. Conparando-se a imagen 33 com a imagem 43 observamos que existem regiões significativamente diferentes em termos de estimativa do minério. Por exemplo podemos destacar a regiões em amarelo e vermelho na borda central da figura 43 bem como no seu canto superior esquerdo respectivamente, onde a intensidade de estimativa foi superior ao encontrado com o método tradicional. Mas também podemos encontrar semelhanças ao observarmos que em grande parte da imagem 43 predomina a cor azul em suas tonalidades clara e escura o que corresponte também a imagem 33 gerada pelo método da krigagem. O fato de um algoritmo relativamente simples apresentar um erro de estimativa menor que o encontrado no método tradicional de krigagem está no fato de utilizarmos vários algoritmos para o cálculo do vizinho mais próximo pois no treinamento dessa máquina de aprendizado vemos que o algoritmo *auto* se mostrou melhor para mapear os dados de ferro e níquel ao passo que o *ball tree* foi mais bem aplicado para o ouro, como mostrado na tabela 9. O resultado mostrado na tabela 19 nos mostra que o *k-NN* errou menos para a estimativa de minério de níquel e esse percentual foi de 21% o que nos mostra uma boa capacidade preditiva para dados espaçados.

6 CONCLUSÃO

O metodologia de AM, de acordo com os resultados obtidos, atende aos critérios de qualidade, no que concerne ao estudo de estimativas de recursos minerais. Tendo em vista que os métodos ensembles se apresentaram como uma ótima alternativa para contribuir nas avaliações das jazidas minerais, em particular pode-se citar os métodos: florestas aleatórias e florestas extra aleatórias os quais já se apresentaram, para os três bancos de dados analisados, como sendo os melhores modelos preditores para estimar a pureza dos teores nas jazidas minerais. O desafio, portanto, reside na questão de operacionalizar um processo de tratamento dessa nova metodologia de estimativa de recursos minerais mais adaptável às necessidades das equipes de engenharia de minas, sejam elas pequenas ou grandes, de forma que o conhecimento empregado no desenvolvimento deste trabalho atenda de modo preciso e eficaz aos requisitos do usuário ou cliente que o aplique em seus projetos emgenharia de mineral. Neste sentido, a pesquisa deste trabalho mostrou-se relevante uma vez que abordou uma demanda bastante relevante da indústria mineral que é a qualidade da estimativa em modelos geológicos de teores de minério, os quais são de fundamental importância em todo o processo de desenvolvimento da mina.

A garantia da qualidade na estimativa de um determinado recurso mineral tem como propósito assegurar a conformidade nos trabalhos de extração do bem mineral avaliado e também fornecer todo o suporte financeiro para a empresa poder operar no mercado de commodities de bens minerais. Logo, qualquer melhoria nesse sendido irá contribuir para a melhoria em todo processo operacional da mina dentro de um curto, médio e longo prazo. Dessa forma, os resultados deste trabalho foram sistematizados de forma a gerar um modelo de referência para a garantia de agilidade e qualidade nas avaliações dos recursos minerais para as indústrias de extração mineral. Esse modelo de referência é atribuído ao método Florestas Extra Aleatórias o qual apresentou erros inferiores ao da krigagem de 35%, 22,5% e 21% para os bancos de dados de Ferro, Ouro e Níquel respectivamente.

Estes modelos de aprendizado computacional visaram primeiramente superar os desafios identificados por meio de revisões sistemáticas de literatura, descritas no Capítulo 2, de forma a garantir que a aplicalçao dos métodos ensembles possibilitem uma mlehor qualidade dentro do contexto de estimativa dos recursos minerais. Para tal, foi apresentado todos os modelos geológicos de teores para o Ferro, Ouro e Níquel os quais apresentaram as suas devidas particularidades dentro do contexto algorítmico em que foram gerados. Por meio desses desses regressores de AM concluí-se que as atividades operacionais da mineração são, significativamente, melhoradas tendo em vista que quanto maior a precisão acerca das informações de distribuição de teores ao longo do corpo mineralizado melhor será para garantir a qualidade no planejamento, avaliação e acompanhamento de toda as fases produtivas da mina, as quais podem ser conduzidas de forma a atender variações de-

correntes das oscilações do mercado de bens minerais, colaboração com o cliente, resposta a mudanças de funcionamento no equipamentos operacionais, retorno de investimento etc. O modelo considera que não existe uma solução padrão para todos os casos, mas um conjunto de possibilidades que podem ser adequadas e readequadas a diferentes contextos, possibilitando que desde pequenas empresas, com poucos recursos financeiros, até grandes empresas, com várias equipes, possam implementar esse modelo de AM e garantir uma melhor qualidade no tratamento de seus recursos minerais.

A tendência atual em relação ao uso de computadores é empregar máquinas computacionais mais rápidas de forma a processar informações com precisão e eficiência, o que garante um melhor desempenho para o treinamento dos dados e a geração imediata dos modelos de AM. Portanto, a implementação de modelos computacionais mais sofisticados repousam em conformidade com a era presente, o que nos encoraja a olhar mais a frente e enxergar novas possibilidades no campo operacional das indústrias minerais.

Dentro desse contexto de AM, o qual é bastante inovador e chega definitivamente para contribuir com o já existente para a estimativa de recursos minerais, pode-se concluir que a aplicação dessas metodologias não se limita apenas as avaliações dos recursos minerais, visto que elas são perfeitamente aplicáveis para encontrar diversos cenários no que concerne ao planejamento de extração dos recursos minerais, o que consequentemente otimiza todo o processo operacional da mina. Portanto, as metodologias de AM se mostram como um ponto de partida de grande importância para a indústria mineral ao trazer uma nova visão dentro da cadeia produtiva dos recursos minerais.

REFERÊNCIAS

ABBAD, G. d. S.; TORRES, C. V. Regressão múltipla stepwise e hierárquica em psicologia organizacional: aplicações, problemas e soluções. SciELO Brasil, 2002.

ALEEM, S.; CAPRETZ, L. F.; AHMED, F. Benchmarking machine learning techniques for software defect detection. *Int. J. Softw. Eng. Appl*, v. 6, n. 3, 2015.

BEKKERMAN, R.; BILENKO, M.; LANGFORD, J. *Scaling up machine learning: Parallel and distributed approaches*. [S.l.]: Cambridge University Press, 2011.

BENNETT, K. P.; MANGASARIAN, O. L. Robust linear programming discrimination of two linearly inseparable sets. *Optimization methods and software*, Taylor & Francis, v. 1, n. 1, p. 23–34, 1992.

BISHOP, C. M. Periodic variables. *Pattern recognition and machine learning*, Springer Science+ Business Media, LLC New York, v. 1, 2006.

BOJARSKI, M.; TESTA, D. D.; DWORAKOWSKI, D.; FIRNER, B.; FLEPP, B.; GOYAL, P.; JACKEL, L. D.; MONFORT, M.; MULLER, U.; ZHANG, J. et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

BREIMAN, L. Bias, variance, and arcing classifiers. Tech. Rep. 460, Statistics Department, University of California, Berkeley . . . , 1996.

BREIMAN, L. Random forests. *Machine learning*, Springer, v. 45, n. 1, p. 5–32, 2001.

BROVELLI, M.; SANSO, F.; VENUTI, G. A discussion on the wiener–kolmogorov prediction principle with easy-to-compute and robust variants. *Journal of Geodesy*, Springer, v. 76, n. 11-12, p. 673–683, 2003.

COOPER, G. F.; ALIFERIS, C. F.; AMBROSINO, R.; ARONIS, J.; BUCHANAN, B. G.; CARUANA, R.; FINE, M. J.; GLYMOUR, C.; GORDON, G.; HANUSA, B. H. et al. An evaluation of machine-learning methods for predicting pneumonia mortality. *Artificial intelligence in medicine*, Elsevier, v. 9, n. 2, p. 107–138, 1997.

CORTES, C.; VAPNIK, V. Support-vector networks. *Machine learning*, Springer, v. 20, n. 3, p. 273–297, 1995.

CRISCI, C.; GHATTAS, B.; PERERA, G. A review of supervised machine learning algorithms and their applications to ecological data. *Ecological Modelling*, v. 240, p. 113–122, 2012. ISSN 0304-3800. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0304380012001081>.

DAVID, M. *Geostatistical ore reserve estimation*. [S.l.]: Elsevier, 2012.

DIETTERICH, T. G. Ensemble methods in machine learning. In: SPRINGER. *International workshop on multiple classifier systems*. [S.l.], 2000. p. 1–15.

DRUCKER, H. Improving regressors using boosting techniques. In: *ICML*. [S.l.: s.n.], 1997. v. 97, p. 107–115.

DUTTA, S.; BANDOPADHYAY, S.; GANGULI, R.; MISRA, D. Machine learning algorithms and their application to ore reserve estimation of sparse and imprecise data. *Journal of Intelligent Learning Systems and Applications*, Scientific Research Publishing, v. 2, n. 02, p. 86, 2010.

FAHLMAN, S. E.; LEBIERE, C. The cascade-correlation learning architecture. In: *Advances in neural information processing systems*. [S.l.: s.n.], 1990. p. 524–532.

FORKES, J. *Geological control and the application of geostatistics to mineral evaluation.* Tese (Doutorado) — Geology, 1982.

FRIEDMAN, J. H. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, JSTOR, p. 1189–1232, 2001.

GEURTS, P.; ERNST, D.; WEHENKEL, L. Extremely randomized trees. *Machine learning*, Springer, v. 63, n. 1, p. 3–42, 2006.

GOOVAERTS, P. Geostatistics in soil science: state-of-the-art and perspectives. *Geoderma*, Elsevier, v. 89, n. 1-2, p. 1–45, 1999.

GOOVAERTS, P. Geostatistical approaches for incorporating elevation into the spatial interpolation of rainfall. *Journal of hydrology*, Elsevier, v. 228, n. 1-2, p. 113–129, 2000.

GRINGARTEN, E.; DEUTSCH, C. V. Teacher's aide variogram interpretation and modeling. *Mathematical Geology*, Springer, v. 33, n. 4, p. 507–534, 2001.

HANSEN, L. K.; SALAMON, P. Neural network ensembles. *IEEE transactions on pattern analysis and machine intelligence*, IEEE, v. 12, n. 10, p. 993–1001, 1990.

JOURNEL, A. G.; HUIJBREGTS, C. J. *Mining geostatistics*. [S.l.]: Academic press London, 1978. v. 600.

KIM, K.-j. Financial time series forecasting using support vector machines. *Neurocomputing*, Elsevier, v. 55, n. 1-2, p. 307–319, 2003.

LIBBRECHT, M. W.; NOBLE, W. S. Machine learning applications in genetics and genomics. *Nature Reviews Genetics*, Nature Publishing Group, v. 16, n. 6, p. 321, 2015.

MARTINHO, C.; GIANNINI, P.; SAWAKUCHI, A.; HESP, P. Morphological and depositional facies of transgressive dunefields in the imbituba-jaguaruna region, santa catarina state, southern brazil. *Journal of Coastal Research*, JSTOR, p. 673–677, 2006.

MATHERON, G. Principles of geostatistics. *Economic geology*, Society of Economic Geologists, v. 58, n. 8, p. 1246–1266, 1963.

MENDES-MOREIRA, J.; SOARES, C.; JORGE, A. M.; SOUSA, J. F. D. Ensemble approaches for regression: A survey. *ACM Computing Surveys (CSUR)*, ACM, v. 45, n. 1, p. 10, 2012.

RANI, P.; LIU, C.; SARKAR, N.; VANMAN, E. An empirical study of machine learning techniques for affect recognition in human–robot interaction. *Pattern Analysis and Applications*, Springer, v. 9, n. 1, p. 58–69, 2006.

RIPLEY, B. D. Flexible non-linear approaches to classification. In: *From Statistics to Neural Networks*. [S.l.]: Springer, 1994. p. 105–126.

SEWELL, M. Ensemble learning. *RN*, v. 11, n. 02, 2008.

TESAURO, G. Temporal difference learning and td-gammon. *Communications of the ACM*, v. 38, n. 3, p. 58–68, 1995.

WANG, G.; HAO, J.; MA, J.; JIANG, H. A comparative assessment of ensemble learning for credit scoring. *Expert systems with applications*, Elsevier, v. 38, n. 1, p. 223–230, 2011.

WU, X.; KUMAR, V.; QUINLAN, J. R.; GHOSH, J.; YANG, Q.; MOTODA, H.; MCLACHLAN, G. J.; NG, A.; LIU, B.; PHILIP, S. Y. et al. Top 10 algorithms in data mining. *Knowledge and information systems*, Springer, v. 14, n. 1, p. 1–37, 2008.

ZOU, H.; HASTIE, T. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, Wiley Online Library, v. 67, n. 2, p. 301–320, 2005.

APÊNDICE A – CÓDIGOS IMPLEMENTADOS EM PYTHON

Listing A.1 – Código em linguagem de programação Python para o treinamento das Máquinas de Aprendizado

```
1
  LINEAR REGRESSION CODE TO Fe
3 import pandas as pd
  import numpy as np
5 from sklearn.linear_model import LinearRegression
  from sklearn.model_selection import cross_val_score
7 from sklearn.model_selection import train_test_split
  from sklearn.model_selection import KFold
9 from sklearn.preprocessing import StandardScaler
  from sklearn.model_selection import GridSearchCV
11
  dataset = pd.read_csv('ferro3.csv')
13 print(dataset.shape)
  print(dataset.dtypes)
15
  LR_Fe = LinearRegression()
17
  array = dataset.values
19 X = array[:,0:3]
  Y = array[:,3]
21 validation_size = 0.20
  seed = 7
23 X_train, X_validation, Y_train, Y_validation = train_test_split(X, Y, test_size=
      validation_size, random_state=seed)

25 num_folds = 10
  seed = 7
27 scoring = 'neg_mean_squared_error'

29 kfold = KFold(n_splits=num_folds, random_state=seed)
  cv_results = cross_val_score(LR_Fe, X_train, Y_train, cv=kfold, scoring=scoring)
31 msg = '%f (%f)' % (cv_results.mean(), cv_results.std())

33 scaler = StandardScaler().fit(X_train)
  rescaledX = scaler.transform(X_train)
35 fit_intercept = [True]
  normalize = [False]
37 copy_X = [True]
  n_jobs = [None]
39
  param_grid = dict(fit_intercept=fit_intercept, normalize=normalize, copy_X=copy_X,
      n_jobs=n_jobs)
41 model = LinearRegression()
  kfold = KFold(n_splits=num_folds, random_state=seed)
43 grid = GridSearchCV(estimator=model, param_grid=param_grid, scoring=scoring, cv=
      kfold)
  grid_result = grid.fit(rescaledX, Y_train)
45
  print('Best: %f using %s' % (grid_result.best_score_, grid_result.best_params_))
```

```
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print('%f (%f) with: %r' % (mean, stdev, param))

scaler = StandardScaler().fit(X_train)
rescaledX = scaler.transform(X_train)
model = LinearRegression(fit_intercept=True, normalize=False, copy_X=True, n_jobs=
    None)
model.fit(rescaledX, Y_train)
from sklearn.metrics import mean_squared_error

rescaledValidationX = scaler.transform(X_validation)
predictions = model.predict(rescaledValidationX)
print(mean_squared_error(Y_validation, predictions))


LINEAR REGRESSION CODE TO Au
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV

dataset = pd.read_csv('ouro.csv')
print(dataset.shape)
print(dataset.dtypes)

LR_Au = LinearRegression()

array = dataset.values
X = array[:,0:3]
Y = array[:,3]
validation_size = 0.20
seed = 7
X_train, X_validation, Y_train, Y_validation = train_test_split(X, Y, test_size=
    validation_size, random_state=seed)

num_folds = 10
seed = 7
scoring = 'neg_mean_squared_error'

kfold = KFold(n_splits=num_folds, random_state=seed)
cv_results = cross_val_score(LR_Au, X_train, Y_train, cv=kfold, scoring=scoring)
msg = '%f (%f)' % (cv_results.mean(), cv_results.std())

scaler = StandardScaler().fit(X_train)
rescaledX = scaler.transform(X_train)
fit_intercept = [True]
normalize = [False]
copy_X = [True]
n_jobs = [None]

```

```
param_grid = dict(fit_intercept=fit_intercept, normalize=normalize, copy_X=copy_X,
    n_jobs=n_jobs)
model = LinearRegression()
kfold = KFold(n_splits=num_folds, random_state=seed)
grid = GridSearchCV(estimator=model, param_grid=param_grid, scoring=scoring, cv=
    kfold)
grid_result = grid.fit(rescaledX, Y_train)

print('Best: %f using %s' % (grid_result.best_score_, grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print('%f (%f) with: %r' % (mean, stdev, param))

scaler = StandardScaler().fit(X_train)
rescaledX = scaler.transform(X_train)
model = LinearRegression(fit_intercept=True, normalize=False, copy_X=True, n_jobs=
    None)
model.fit(rescaledX, Y_train)
from sklearn.metrics import mean_squared_error

rescaledValidationX = scaler.transform(X_validation)
predictions = model.predict(rescaledValidationX)
print(mean_squared_error(Y_validation, predictions))


LINEAR REGRESSION CODE TO Ni
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV

dataset = pd.read_csv('niquel.csv')
print(dataset.shape)
print(dataset.dtypes)

LR_Ni = LinearRegression()

array = dataset.values
X = array[:,0:3]
Y = array[:,3]
validation_size = 0.20
seed = 7
X_train, X_validation, Y_train, Y_validation = train_test_split(X, Y, test_size=
    validation_size, random_state=seed)

num_folds = 10
seed = 7
scoring = 'neg_mean_squared_error'

kfold = KFold(n_splits=num_folds, random_state=seed)
cv_results = cross_val_score(LR_Ni, X_train, Y_train, cv=kfold, scoring=scoring)
```

```
msg = '%f (%f)' % (cv_results.mean(), cv_results.std())

scaler = StandardScaler().fit(X_train)
rescaledX = scaler.transform(X_train)
fit_intercept = [True]
normalize = [False]
copy_X = [True]
n_jobs = [None]

param_grid = dict(fit_intercept=fit_intercept, normalize=normalize, copy_X=copy_X,
    n_jobs=n_jobs)
model = LinearRegression()
kfold = KFold(n_splits=num_folds, random_state=seed)
grid = GridSearchCV(estimator=model, param_grid=param_grid, scoring=scoring, cv=
    kfold)
grid_result = grid.fit(rescaledX, Y_train)

print('Best: %f using %s' % (grid_result.best_score_, grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print('%f (%f) with: %r' % (mean, stdev, param))

scaler = StandardScaler().fit(X_train)
rescaledX = scaler.transform(X_train)
model = LinearRegression(fit_intercept=True, normalize=False, copy_X=True, n_jobs=
    None)
model.fit(rescaledX, Y_train)
from sklearn.metrics import mean_squared_error

rescaledValidationX = scaler.transform(X_validation)
predictions = model.predict(rescaledValidationX)
print(mean_squared_error(Y_validation, predictions))


LASSO REGRESSION CODE TO Fe
import pandas as pd
import numpy as np
from sklearn.linear_model import Lasso
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV

dataset = pd.read_csv('ferro3.csv')
print(dataset.shape)
print(dataset.dtypes)

LASSO_Fe = Lasso()

array = dataset.values
X = array[:,0:3]
Y = array[:,3]
validation_size = 0.20
seed = 7
```

```
X_train, X_validation, Y_train, Y_validation = train_test_split(X, Y, test_size=validation_size, random_state=seed)

num_folds = 10
seed = 7
scoring = 'neg_mean_squared_error'

kfold = KFold(n_splits=num_folds, random_state=seed)
cv_results = cross_val_score(LASSO_Fe, X_train, Y_train, cv=kfold, scoring=scoring)
msg = '%f (%f)' % (cv_results.mean(), cv_results.std())

scaler = StandardScaler().fit(X_train)
rescaledX = scaler.transform(X_train)
fit_intercept = [True]
normalize = [False]
copy_X = [True]
max_iter = [100, 1000, 10000, 100000]
tol = [0.001, 0.0001, 0.00001, 0.000001]
warm_start = [False]
positive = [False]
selection = ['cyclic']
param_grid = dict(fit_intercept=fit_intercept, normalize=normalize, copy_X=copy_X, max_iter=max_iter, tol=tol,
                  warm_start=warm_start, positive=positive, selection=selection)
model = Lasso()
kfold = KFold(n_splits=num_folds, random_state=seed)
grid = GridSearchCV(estimator=model, param_grid=param_grid, scoring=scoring, cv=kfold)
grid_result = grid.fit(rescaledX, Y_train)

print('Best: %f using %s' % (grid_result.best_score_, grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print('%f (%f) with: %r' % (mean, stdev, param))

scaler = StandardScaler().fit(X_train)
rescaledX = scaler.transform(X_train)
model = Lasso(fit_intercept=True, normalize=False, copy_X=True, max_iter=100, positive=False, selection='cyclic',
              tol=0.001, warm_start=False)
model.fit(rescaledX, Y_train)
from sklearn.metrics import mean_squared_error

rescaledValidationX = scaler.transform(X_validation)
predictions = model.predict(rescaledValidationX)
print(mean_squared_error(Y_validation, predictions))


LASSO REGRESSION CODE TO Au
import pandas as pd
import numpy as np
from sklearn.linear_model import Lasso
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
```

```
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV

dataset = pd.read_csv('ouro.csv')
print(dataset.shape)
print(dataset.dtypes)

LASSO_Au = Lasso()

array = dataset.values
X = array[:,0:3]
Y = array[:,3]
validation_size = 0.20
seed = 7
X_train, X_validation, Y_train, Y_validation = train_test_split(X, Y, test_size=
    validation_size, random_state=seed)

num_folds = 10
seed = 7
scoring = 'neg_mean_squared_error'

kfold = KFold(n_splits=num_folds, random_state=seed)
cv_results = cross_val_score(LASSO_Au, X_train, Y_train, cv=kfold, scoring=scoring)
msg = '%f (%f)' % (cv_results.mean(), cv_results.std())

scaler = StandardScaler().fit(X_train)
rescaledX = scaler.transform(X_train)
fit_intercept = [True]
normalize = [False]
copy_X = [True]
max_iter = [100, 1000, 10000, 100000]
tol = [0.001, 0.0001, 0.00001, 0.000001]
warm_start = [False]
positive = [False]
selection = ['cyclic']
param_grid = dict(fit_intercept=fit_intercept, normalize=normalize, copy_X=copy_X,
    max_iter=max_iter, tol=tol,
                  warm_start=warm_start, positive=positive, selection=selection)
model = Lasso()
kfold = KFold(n_splits=num_folds, random_state=seed)
grid = GridSearchCV(estimator=model, param_grid=param_grid, scoring=scoring, cv=
    kfold)
grid_result = grid.fit(rescaledX, Y_train)

print('Best: %f (%F) using %s' % (grid_result.best_score_, grid_result.best_params_)
    )
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print('%f (%f) with: %r' % (mean, stdev, param))

scaler = StandardScaler().fit(X_train)
rescaledX = scaler.transform(X_train)
model = Lasso(fit_intercept=True, normalize=False, copy_X=True, max_iter=100,
    positive=False, selection='cyclic',
              tol=0.001, warm_start=False)
```

```
model.fit(rescaledX, Y_train)
from sklearn.metrics import mean_squared_error

rescaledValidationX = scaler.transform(X_validation)
predictions = model.predict(rescaledValidationX)
print(mean_squared_error(Y_validation, predictions))


LASSO REGRESSION CODE TO Ni
import pandas as pd
import numpy as np
from sklearn.linear_model import Lasso
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV

dataset = pd.read_csv('niquel.csv')
print(dataset.shape)
print(dataset.dtypes)

LASSO_Ni = Lasso()

array = dataset.values
X = array[:,0:3]
Y = array[:,3]
validation_size = 0.20
seed = 7
X_train, X_validation, Y_train, Y_validation = train_test_split(X, Y, test_size=
    validation_size, random_state=seed)

num_folds = 10
seed = 7
scoring = 'neg_mean_squared_error'

kfold = KFold(n_splits=num_folds, random_state=seed)
cv_results = cross_val_score(LASSO_Ni, X_train, Y_train, cv=kfold, scoring=scoring)
msg = '%f (%f)' % (cv_results.mean(), cv_results.std())

scaler = StandardScaler().fit(X_train)
rescaledX = scaler.transform(X_train)
fit_intercept = [True]
normalize = [False]
copy_X = [True]
max_iter = [100, 1000, 10000, 100000]
tol = [0.001, 0.0001, 0.00001, 0.000001]
warm_start = [False]
positive = [False]
selection = ['cyclic']
param_grid = dict(fit_intercept=fit_intercept, normalize=normalize, copy_X=copy_X,
    max_iter=max_iter, tol=tol,
                  warm_start=warm_start, positive=positive, selection=selection)
model = Lasso()
kfold = KFold(n_splits=num_folds, random_state=seed)
```

```
grid = GridSearchCV(estimator=model, param_grid=param_grid, scoring=scoring, cv=kfold)
grid_result = grid.fit(rescaledX, Y_train)

print('Best: %f using %s' % (grid_result.best_score_, grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print('%f (%f) with: %r' % (mean, stdev, param))

scaler = StandardScaler().fit(X_train)
rescaledX = scaler.transform(X_train)
model = Lasso(fit_intercept=True, normalize=False, copy_X=True, max_iter=100, positive=False, selection='cyclic',
              tol=0.001, warm_start=False)
model.fit(rescaledX, Y_train)
from sklearn.metrics import mean_squared_error

rescaledValidationX = scaler.transform(X_validation)
predictions = model.predict(rescaledValidationX)
print(mean_squared_error(Y_validation, predictions))


ELASTICNET REGRESSION CODE TO Fe
import pandas as pd
import numpy as np
from sklearn.linear_model import ElasticNet
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV

dataset = pd.read_csv('ferro3.csv')
print(dataset.shape)
print(dataset.dtypes)

EL = ElasticNet()

array = dataset.values
X = array[:,0:3]
Y = array[:,3]
validation_size = 0.20
seed = 7
X_train, X_validation, Y_train, Y_validation = train_test_split(X, Y, test_size=validation_size, random_state=seed)

num_folds = 10
seed = 7
scoring = 'neg_mean_squared_error'

kfold = KFold(n_splits=num_folds, random_state=seed)
cv_results = cross_val_score(EL, X_train, Y_train, cv=kfold, scoring=scoring)
msg = '%f (%f)' % (cv_results.mean(), cv_results.std())

scaler = StandardScaler().fit(X_train)
```

```
rescaledX = scaler.transform(X_train)
alpha = [1.0]
l1_ratio = [0.5]
fit_intercept = [True]
normalize = [False]
copy_X = [True]
max_iter = [100, 1000, 10000, 100000]
tol = [0.001, 0.0001, 0.00001, 0.000001]
warm_start = [False]
positive = [False]
selection = ['cyclic']
param_grid = dict(fit_intercept=fit_intercept, normalize=normalize, copy_X=copy_X,
    max_iter=max_iter, tol=tol,
                  warm_start=warm_start, positive=positive, selection=selection,
                      alpha=alpha, l1_ratio=l1_ratio)

model = ElasticNet()
kfold = KFold(n_splits=num_folds, random_state=seed)
grid = GridSearchCV(estimator=model, param_grid=param_grid, scoring=scoring, cv=
    kfold)
grid_result = grid.fit(rescaledX, Y_train)

print('Best: %f using %s' % (grid_result.best_score_, grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print('%f (%f) with: %r' % (mean, stdev, param))

scaler = StandardScaler().fit(X_train)
rescaledX = scaler.transform(X_train)
model = ElasticNet(fit_intercept=True, normalize=False, copy_X=True, max_iter=100,
    positive=False, selection='cyclic',
              tol=0.001, warm_start=False, alpha=1.0, l1_ratio=0.5)
model.fit(rescaledX, Y_train)
from sklearn.metrics import mean_squared_error

rescaledValidationX = scaler.transform(X_validation)
predictions = model.predict(rescaledValidationX)
print(mean_squared_error(Y_validation, predictions))


ELASTICNET REGRESSION CODE TO Au
import pandas as pd
import numpy as np
from sklearn.linear_model import ElasticNet
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV

dataset = pd.read_csv('ouro.csv')
print(dataset.shape)
print(dataset.dtypes)

EL_Au = ElasticNet()
```

```
array = dataset.values
X = array[:,0:3]
Y = array[:,3]
validation_size = 0.20
seed = 7
X_train, X_validation, Y_train, Y_validation = train_test_split(X, Y, test_size=
    validation_size, random_state=seed)

num_folds = 10
seed = 7
scoring = 'neg_mean_squared_error'

kfold = KFold(n_splits=num_folds, random_state=seed)
cv_results = cross_val_score(EL_Au, X_train, Y_train, cv=kfold, scoring=scoring)
msg = '%f (%f)' % (cv_results.mean(), cv_results.std())

scaler = StandardScaler().fit(X_train)
rescaledX = scaler.transform(X_train)
alpha = [1.0]
l1_ratio = [0.5]
fit_intercept = [True]
normalize = [False]
copy_X = [True]
max_iter = [100, 1000, 10000, 100000]
tol = [0.001, 0.0001, 0.00001, 0.000001]
warm_start = [False]
positive = [False]
selection = ['cyclic']
param_grid = dict(fit_intercept=fit_intercept, normalize=normalize, copy_X=copy_X,
    max_iter=max_iter, tol=tol,
                  warm_start=warm_start, positive=positive, selection=selection,
                      alpha=alpha, l1_ratio=l1_ratio)

model = ElasticNet()
kfold = KFold(n_splits=num_folds, random_state=seed)
grid = GridSearchCV(estimator=model, param_grid=param_grid, scoring=scoring, cv=
    kfold)
grid_result = grid.fit(rescaledX, Y_train)

print('Best: %f using %s' % (grid_result.best_score_, grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print('%f (%f) with: %r' % (mean, stdev, param))

scaler = StandardScaler().fit(X_train)
rescaledX = scaler.transform(X_train)
model = ElasticNet(fit_intercept=True, normalize=False, copy_X=True, max_iter=100,
    positive=False, selection='cyclic',
              tol=0.001, warm_start=False, alpha=1.0, l1_ratio=0.5)
model.fit(rescaledX, Y_train)
from sklearn.metrics import mean_squared_error

rescaledValidationX = scaler.transform(X_validation)
predictions = model.predict(rescaledValidationX)
```

```
print(mean_squared_error(Y_validation, predictions))


ELASTICNET REGRESSION CODE TO Ni
import pandas as pd
import numpy as np
from sklearn.linear_model import ElasticNet
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV

dataset = pd.read_csv('niquel.csv')
print(dataset.shape)
print(dataset.dtypes)

EN_Ni = ElasticNet()

array = dataset.values
X = array[:,0:3]
Y = array[:,3]
validation_size = 0.20
seed = 7
X_train, X_validation, Y_train, Y_validation = train_test_split(X, Y, test_size=
    validation_size, random_state=seed)

num_folds = 10
seed = 7
scoring = 'neg_mean_squared_error'

kfold = KFold(n_splits=num_folds, random_state=seed)
cv_results = cross_val_score(EN_Ni, X_train, Y_train, cv=kfold, scoring=scoring)
msg = '%f (%f)' % (cv_results.mean(), cv_results.std())

scaler = StandardScaler().fit(X_train)
rescaledX = scaler.transform(X_train)
alpha = [1.0]
l1_ratio = [0.5]
fit_intercept = [True]
normalize = [False]
copy_X = [True]
max_iter = [100, 1000, 10000, 100000]
tol = [0.001, 0.0001, 0.00001, 0.000001]
warm_start = [False]
positive = [False]
selection = ['cyclic']
param_grid = dict(fit_intercept=fit_intercept, normalize=normalize, copy_X=copy_X,
    max_iter=max_iter, tol=tol,
                  warm_start=warm_start, positive=positive, selection=selection,
                      alpha=alpha, l1_ratio=l1_ratio)

model = ElasticNet()
kfold = KFold(n_splits=num_folds, random_state=seed)
grid = GridSearchCV(estimator=model, param_grid=param_grid, scoring=scoring, cv=
    kfold)
grid_result = grid.fit(rescaledX, Y_train)
```

```
581
print('Best: %f using %s' % (grid_result.best_score_, grid_result.best_params_))
583 means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
585 params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
587     print('%f (%f) with: %r' % (mean, stdev, param))

589 scaler = StandardScaler().fit(X_train)
rescaledX = scaler.transform(X_train)
591 model = ElasticNet(fit_intercept=True, normalize=False, copy_X=True, max_iter=100,
    positive=False, selection='cyclic',
              tol=0.001, warm_start=False, alpha=1.0, l1_ratio=0.5)
593 model.fit(rescaledX, Y_train)
from sklearn.metrics import mean_squared_error
595
rescaledValidationX = scaler.transform(X_validation)
597 predictions = model.predict(rescaledValidationX)
print(mean_squared_error(Y_validation, predictions))
599

601 CART REGRESSION CODE TO Fe
import pandas as pd
603 import numpy as np
from sklearn.tree import DecisionTreeRegressor
605 from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
607 from sklearn.model_selection import KFold
from sklearn.preprocessing import StandardScaler
609 from sklearn.model_selection import GridSearchCV

611 dataset = pd.read_csv('ferro3.csv')
print(dataset.shape)
613 print(dataset.dtypes)

615 CART_Fe = DecisionTreeRegressor()

617 array = dataset.values
X = array[:,0:3]
619 Y = array[:,3]
validation_size = 0.20
621 seed = 7
X_train, X_validation, Y_train, Y_validation = train_test_split(X, Y, test_size=
    validation_size, random_state=seed)
623
num_folds = 10
625 seed = 7
scoring = 'neg_mean_squared_error'
627
kfold = KFold(n_splits=num_folds, random_state=seed)
629 cv_results = cross_val_score(CART_Fe, X_train, Y_train, cv=kfold, scoring=scoring)
msg = '%f (%f)' % (cv_results.mean(), cv_results.std())
631
scaler = StandardScaler().fit(X_train)
633 rescaledX = scaler.transform(X_train)
criterion = ['mse']
635 splitter = ['best']
```

```
max_depth = [None]
min_samples_split = [2,3,4,5,6,7,8]
min_samples_leaf = [1,2,3,4,5,6]
min_weight_fraction_leaf = [0,0.5]
max_features = ['auto','sqrt','log2']
random_state = [None]
max_leaf_nodes = [None]
param_grid = dict(criterion=criterion, splitter=splitter, max_depth=max_depth,
    min_samples_split=min_samples_split,
                  min_samples_leaf=min_samples_leaf, min_weight_fraction_leaf=
                      min_weight_fraction_leaf,
                   max_features=max_features,random_state=random_state,
                       max_leaf_nodes=max_leaf_nodes)
model = DecisionTreeRegressor()
kfold = KFold(n_splits=num_folds, random_state=seed)
grid = GridSearchCV(estimator=model, param_grid=param_grid, scoring=scoring, cv=
    kfold)
grid_result = grid.fit(rescaledX, Y_train)

print('Best: %f using %s' % (grid_result.best_score_, grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print('%f (%f) with: %r' % (mean, stdev, param))

scaler = StandardScaler().fit(X_train)
rescaledX = scaler.transform(X_train)
model = DecisionTreeRegressor(criterion='mse',splitter='best', max_depth=None,
    min_samples_split=5, min_samples_leaf=5,
                              min_weight_fraction_leaf=0, max_features='log2',
                                  random_state=None, max_leaf_nodes=None)
model.fit(rescaledX, Y_train)
from sklearn.metrics import mean_squared_error

rescaledValidationX = scaler.transform(X_validation)
predictions = model.predict(rescaledValidationX)
print(mean_squared_error(Y_validation, predictions))


CART REGRESSION CODE TO Au
import pandas as pd
import numpy as np
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV

dataset = pd.read_csv('ouro.csv')
print(dataset.shape)
print(dataset.dtypes)

CART_Au = DecisionTreeRegressor()

array = dataset.values
```

```
X = array[:,0:3]
Y = array[:,3]
validation_size = 0.20
seed = 7
X_train, X_validation, Y_train, Y_validation = train_test_split(X, Y, test_size=
    validation_size, random_state=seed)

num_folds = 10
seed = 7
scoring = 'neg_mean_squared_error'

kfold = KFold(n_splits=num_folds, random_state=seed)
cv_results = cross_val_score(CART_Au, X_train, Y_train, cv=kfold, scoring=scoring)
msg = '%f (%f)' % (cv_results.mean(), cv_results.std())

scaler = StandardScaler().fit(X_train)
rescaledX = scaler.transform(X_train)
criterion = ['mse']
splitter = ['best']
max_depth = [None]
min_samples_split = [2,3,4,5,6,7,8]
min_samples_leaf = [1,2,3,4,5,6]
min_weight_fraction_leaf = [0,0.5]
max_features = ['auto','sqrt','log2']
random_state = [None]
max_leaf_nodes = [None]
param_grid = dict(criterion=criterion, splitter=splitter, max_depth=max_depth,
    min_samples_split=min_samples_split,
                  min_samples_leaf=min_samples_leaf, min_weight_fraction_leaf=
                      min_weight_fraction_leaf,
                   max_features=max_features,random_state=random_state,
                       max_leaf_nodes=max_leaf_nodes)
model = DecisionTreeRegressor()
kfold = KFold(n_splits=num_folds, random_state=seed)
grid = GridSearchCV(estimator=model, param_grid=param_grid, scoring=scoring, cv=
    kfold)
grid_result = grid.fit(rescaledX, Y_train)

print('Best: %f using %s' % (grid_result.best_score_, grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print('%f (%f) with: %r' % (mean, stdev, param))

scaler = StandardScaler().fit(X_train)
rescaledX = scaler.transform(X_train)
model = DecisionTreeRegressor(criterion='mse',splitter='best', max_depth=None,
    min_samples_split=6, min_samples_leaf=1,
                              min_weight_fraction_leaf=0, max_features='log2',
                                  random_state=None, max_leaf_nodes=None)
model.fit(rescaledX, Y_train)
from sklearn.metrics import mean_squared_error

rescaledValidationX = scaler.transform(X_validation)
predictions = model.predict(rescaledValidationX)
print(mean_squared_error(Y_validation, predictions))
```

```
737


739 CART REGRESSION CODE TO Ni
    import pandas as pd
741 import numpy as np
    from sklearn.tree import DecisionTreeRegressor
743 from sklearn.model_selection import cross_val_score
    from sklearn.model_selection import train_test_split
745 from sklearn.model_selection import KFold
    from sklearn.preprocessing import StandardScaler
747 from sklearn.model_selection import GridSearchCV

749 dataset = pd.read_csv('niquel.csv')
    print(dataset.shape)
751 print(dataset.dtypes)

753 CART_Ni = DecisionTreeRegressor()

755 array = dataset.values
    X = array[:,0:3]
757 Y = array[:,3]
    validation_size = 0.20
759 seed = 7
    X_train, X_validation, Y_train, Y_validation = train_test_split(X, Y, test_size=
        validation_size, random_state=seed)
761
    num_folds = 10
763 seed = 7
    scoring = 'neg_mean_squared_error'
765
    kfold = KFold(n_splits=num_folds, random_state=seed)
767 cv_results = cross_val_score(CART_Ni, X_train, Y_train, cv=kfold, scoring=scoring)
    msg = '%f (%f)' % (cv_results.mean(), cv_results.std())
769
    scaler = StandardScaler().fit(X_train)
771 rescaledX = scaler.transform(X_train)
    criterion = ['mse']
773 splitter = ['best']
    max_depth = [None]
775 min_samples_split = [2,3,4,5,6,7,8]
    min_samples_leaf = [1,2,3,4,5,6]
777 min_weight_fraction_leaf = [0,0.5]
    max_features = ['auto','sqrt','log2']
779 random_state = [None]
    max_leaf_nodes = [None]
781 param_grid = dict(criterion=criterion, splitter=splitter, max_depth=max_depth,
        min_samples_split=min_samples_split,
                    min_samples_leaf=min_samples_leaf, min_weight_fraction_leaf=
                        min_weight_fraction_leaf,
783                  max_features=max_features,random_state=random_state,
                        max_leaf_nodes=max_leaf_nodes)
    model = DecisionTreeRegressor()
785 kfold = KFold(n_splits=num_folds, random_state=seed)
    grid = GridSearchCV(estimator=model, param_grid=param_grid, scoring=scoring, cv=
        kfold)
787 grid_result = grid.fit(rescaledX, Y_train)
```

```
print('Best: %f using %s' % (grid_result.best_score_, grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print('%f (%f) with: %r' % (mean, stdev, param))

scaler = StandardScaler().fit(X_train)
rescaledX = scaler.transform(X_train)
model = DecisionTreeRegressor(criterion='mse',splitter='best', max_depth=None,
    min_samples_split=7, min_samples_leaf=3,
                              min_weight_fraction_leaf=0, max_features='auto',
                                  random_state=None, max_leaf_nodes=None)
model.fit(rescaledX, Y_train)
from sklearn.metrics import mean_squared_error

rescaledValidationX = scaler.transform(X_validation)
predictions = model.predict(rescaledValidationX)
print(mean_squared_error(Y_validation, predictions))


SVR REGRESSION CODE TO Fe
import pandas as pd
import numpy as np
from sklearn.svm import SVR
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV
import math

dataset = pd.read_csv('ferro3.csv')
print(dataset.shape)
print(dataset.dtypes)

SVR_Fe = SVR()

array = dataset.values
X = array[:,0:3]
Y = array[:,3]
validation_size = 0.20
seed = 7
X_train, X_validation, Y_train, Y_validation = train_test_split(X, Y, test_size=
    validation_size, random_state=seed)

num_folds = 10
seed = 7
scoring = 'neg_mean_squared_error'

kfold = KFold(n_splits=num_folds, random_state=seed)
cv_results = cross_val_score(SVR_Fe, X_train, Y_train, cv=kfold, scoring=scoring)
msg = '%f (%f)' % (cv_results.mean(), cv_results.std())

scaler = StandardScaler().fit(X_train)
rescaledX = scaler.transform(X_train)
kernel = ['poly','rbf','sigmoid']
```

```
degree = [1,2,3,4]
C = [2**-6, 2**-3, 2**0, 2**3, 2**6]
tol = [0.01, 0.001, 0.0001]

param_grid = dict(kernel=kernel, degree=degree, C=C, tol=tol)
model = SVR()
kfold = KFold(n_splits=num_folds, random_state=seed)
grid = GridSearchCV(estimator=model, param_grid=param_grid, scoring=scoring, cv=
    kfold)
grid_result = grid.fit(rescaledX, Y_train)

print('Best: %f using %s' % (grid_result.best_score_, grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print('%f (%f) with: %r' % (mean, stdev, param))

scaler = StandardScaler().fit(X_train)
rescaledX = scaler.transform(X_train)
model = SVR(kernel='rbf',C=1,degree=1,tol=0.0001)
model.fit(rescaledX, Y_train)
from sklearn.metrics import mean_squared_error

rescaledValidationX = scaler.transform(X_validation)
predictions = model.predict(rescaledValidationX)
print(mean_squared_error(Y_validation, predictions))


SVR REGRESSION CODE TO Au
import pandas as pd
import numpy as np
from sklearn.svm import SVR
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV
import math

dataset = pd.read_csv('ouro.csv')
print(dataset.shape)
print(dataset.dtypes)

SVR_Au = SVR()

array = dataset.values
X = array[:,0:3]
Y = array[:,3]
validation_size = 0.20
seed = 7
X_train, X_validation, Y_train, Y_validation = train_test_split(X, Y, test_size=
    validation_size, random_state=seed)

num_folds = 10
seed = 7
scoring = 'neg_mean_squared_error'
```

```
kfold = KFold(n_splits=num_folds, random_state=seed)
cv_results = cross_val_score(SVR_Au, X_train, Y_train, cv=kfold, scoring=scoring)
msg = '%f (%f)' % (cv_results.mean(), cv_results.std())

scaler = StandardScaler().fit(X_train)
rescaledX = scaler.transform(X_train)
kernel = ['poly','rbf','sigmoid']
degree = [1,2,3,4]
C = [2**-6, 2**-3, 2**0, 2**3, 2**6]
tol = [0.01, 0.001, 0.0001]

param_grid = dict(kernel=kernel, degree=degree, C=C, tol=tol)
model = SVR()
kfold = KFold(n_splits=num_folds, random_state=seed)
grid = GridSearchCV(estimator=model, param_grid=param_grid, scoring=scoring, cv=
    kfold)
grid_result = grid.fit(rescaledX, Y_train)

print('Best: %f using %s' % (grid_result.best_score_, grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print('%f (%f) with: %r' % (mean, stdev, param))

scaler = StandardScaler().fit(X_train)
rescaledX = scaler.transform(X_train)
model = SVR(kernel='rbf',C=64,degree=1,tol=0.0001)
model.fit(rescaledX, Y_train)
from sklearn.metrics import mean_squared_error

rescaledValidationX = scaler.transform(X_validation)
predictions = model.predict(rescaledValidationX)
print(mean_squared_error(Y_validation, predictions))


SVR REGRESSION CODE TO Ni
import pandas as pd
import numpy as np
from sklearn.svm import SVR
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV
import math

dataset = pd.read_csv('niquel.csv')
print(dataset.shape)
print(dataset.dtypes)

SVR_Ni = SVR()

array = dataset.values
X = array[:,0:3]
Y = array[:,3]
```

```
validation_size = 0.20
seed = 7
X_train, X_validation, Y_train, Y_validation = train_test_split(X, Y, test_size=
    validation_size, random_state=seed)

num_folds = 10
seed = 7
scoring = 'neg_mean_squared_error'

kfold = KFold(n_splits=num_folds, random_state=seed)
cv_results = cross_val_score(SVR_Ni, X_train, Y_train, cv=kfold, scoring=scoring)
msg = '%f (%f)' % (cv_results.mean(), cv_results.std())

scaler = StandardScaler().fit(X_train)
rescaledX = scaler.transform(X_train)
kernel = ['poly','rbf','sigmoid']
degree = [1,2,3,4]
C = [2**-6, 2**-3, 2**0, 2**3, 2**6]
tol = [0.01, 0.001, 0.0001]

param_grid = dict(kernel=kernel, degree=degree, C=C, tol=tol)
model = SVR()
kfold = KFold(n_splits=num_folds, random_state=seed)
grid = GridSearchCV(estimator=model, param_grid=param_grid, scoring=scoring, cv=
    kfold)
grid_result = grid.fit(rescaledX, Y_train)

print('Best: %f using %s' % (grid_result.best_score_, grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print('%f (%f) with: %r' % (mean, stdev, param))

scaler = StandardScaler().fit(X_train)
rescaledX = scaler.transform(X_train)
model = SVR(kernel='rbf',C=64,degree=1,tol=0.001)
model.fit(rescaledX, Y_train)
from sklearn.metrics import mean_squared_error

rescaledValidationX = scaler.transform(X_validation)
predictions = model.predict(rescaledValidationX)
print(mean_squared_error(Y_validation, predictions))


KNN REGRESSION CODE TO Fe
import pandas as pd
import numpy as np
from sklearn.neighbors import KNeighborsRegressor
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV

dataset = pd.read_csv('ferro3.csv')
print(dataset.shape)
```

```
print(dataset.dtypes)

KNN_Fe = KNeighborsRegressor()

array = dataset.values
X = array[:,0:3]
Y = array[:,3]
validation_size = 0.20
seed = 7
X_train, X_validation, Y_train, Y_validation = train_test_split(X, Y, test_size=
    validation_size, random_state=seed)

num_folds = 10
seed = 7
scoring = 'neg_mean_squared_error'

kfold = KFold(n_splits=num_folds, random_state=seed)
cv_results = cross_val_score(KNN_Fe, X_train, Y_train, cv=kfold, scoring=scoring)
msg = '%f (%f)' % (cv_results.mean(), cv_results.std())

scaler = StandardScaler().fit(X_train)
rescaledX = scaler.transform(X_train)
k_values = np.array([1,3,5,7,9,11,13,15,17,19,21])
weights = ['uniform','distance']
algorithm = ['auto','ball_tree', 'kd_tree', 'brute']
leaf_size = [10,20,30]
param_grid = dict(n_neighbors=k_values, weights=weights, algorithm=algorithm,
    leaf_size=leaf_size)
model = KNeighborsRegressor()
kfold = KFold(n_splits=num_folds, random_state=seed)
grid = GridSearchCV(estimator=model, param_grid=param_grid, scoring=scoring, cv=
    kfold)
grid_result = grid.fit(rescaledX, Y_train)

print('Best: %f using %s' % (grid_result.best_score_, grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print('%f (%f) with: %r' % (mean, stdev, param))

scaler = StandardScaler().fit(X_train)
rescaledX = scaler.transform(X_train)
model = KNeighborsRegressor(n_neighbors=3, weights='distance', algorithm='auto',
    leaf_size=30 )
model.fit(rescaledX, Y_train)
from sklearn.metrics import mean_squared_error

rescaledValidationX = scaler.transform(X_validation)
predictions = model.predict(rescaledValidationX)
print(mean_squared_error(Y_validation, predictions))


KNN REGRESSION CODE TO Au
import pandas as pd
import numpy as np
from sklearn.neighbors import KNeighborsRegressor
```

```
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV

dataset = pd.read_csv('ouro.csv')
print(dataset.shape)
print(dataset.dtypes)

KNN_Au = KNeighborsRegressor()

array = dataset.values
X = array[:,0:3]
Y = array[:,3]
validation_size = 0.20
seed = 7
X_train, X_validation, Y_train, Y_validation = train_test_split(X, Y, test_size=
    validation_size, random_state=seed)

num_folds = 10
seed = 7
scoring = 'neg_mean_squared_error'

kfold = KFold(n_splits=num_folds, random_state=seed)
cv_results = cross_val_score(KNN_Au, X_train, Y_train, cv=kfold, scoring=scoring)
msg = '%f (%f)' % (cv_results.mean(), cv_results.std())

k_values = np.array([1,3,5,7,9,11,13,15,17,19,21])
weights = ['uniform','distance']
algorithm = ['auto','ball_tree', 'kd_tree', 'brute']
leaf_size = [10,20,30]
param_grid = dict(n_neighbors=k_values, weights=weights, algorithm=algorithm,
    leaf_size=leaf_size)
model = KNeighborsRegressor()
kfold = KFold(n_splits=num_folds, random_state=seed)
grid = GridSearchCV(estimator=model, param_grid=param_grid, scoring=scoring, cv=
    kfold)
grid_result = grid.fit(X_train, Y_train)

print('Best: %f using %s' % (grid_result.best_score_, grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print('%f (%f) with: %r' % (mean, stdev, param))

model = KNeighborsRegressor(n_neighbors=7, weights='distance', algorithm='ball_tree'
    , leaf_size=10 )
model.fit(X_train, Y_train)
from sklearn.metrics import mean_squared_error

predictions = model.predict(X_validation)
print(mean_squared_error(Y_validation, predictions))


KNN REGRESSION CODE TO Ni
```

```
import pandas as pd
import numpy as np
from sklearn.neighbors import KNeighborsRegressor
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV

dataset = pd.read_csv('niquel.csv')
print(dataset.shape)
print(dataset.dtypes)

KNN_Ni = KNeighborsRegressor()

array = dataset.values
X = array[:,0:3]
Y = array[:,3]
validation_size = 0.20
seed = 7
X_train, X_validation, Y_train, Y_validation = train_test_split(X, Y, test_size=
    validation_size, random_state=seed)

num_folds = 10
seed = 7
scoring = 'neg_mean_squared_error'

kfold = KFold(n_splits=num_folds, random_state=seed)
cv_results = cross_val_score(KNN_Ni, X_train, Y_train, cv=kfold, scoring=scoring)
msg = '%f (%f)' % (cv_results.mean(), cv_results.std())

k_values = np.array([1,3,5,7,9,11,13,15,17,19,21])
weights = ['uniform','distance']
algorithm = ['auto','ball_tree', 'kd_tree', 'brute']
leaf_size = [10,20,30]
param_grid = dict(n_neighbors=k_values, weights=weights, algorithm=algorithm,
    leaf_size=leaf_size)
model = KNeighborsRegressor()
kfold = KFold(n_splits=num_folds, random_state=seed)
grid = GridSearchCV(estimator=model, param_grid=param_grid, scoring=scoring, cv=
    kfold)
grid_result = grid.fit(X_train, Y_train)

print('Best: %f using %s' % (grid_result.best_score_, grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print('%f (%f) with: %r' % (mean, stdev, param))

from sklearn.metrics import mean_squared_error
scaler = StandardScaler().fit(X_train)
rescaledX = scaler.transform(X_train)
model = KNeighborsRegressor(n_neighbors=3, weights='distance', algorithm='auto',
    leaf_size=20 )
model.fit(rescaledX, Y_train)
```

```
rescaledValidationX = scaler.transform(X_validation)
predictions = model.predict(rescaledValidationX)
print(mean_squared_error(Y_validation, predictions))


ADABOOST REGRESSION CODE TO Fe
import pandas as pd
import numpy as np
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import AdaBoostRegressor
import math

dataset = pd.read_csv('ferro3.csv')
print(dataset.shape)
print(dataset.dtypes)

ADABOOSTR_Fe = AdaBoostRegressor()

array = dataset.values
X = array[:,0:3]
Y = array[:,3]
validation_size = 0.20
seed = 7
X_train, X_validation, Y_train, Y_validation = train_test_split(X, Y, test_size=
    validation_size, random_state=seed)

num_folds = 10
seed = 7
scoring = 'neg_mean_squared_error'

kfold = KFold(n_splits=num_folds, random_state=seed)
cv_results = cross_val_score(ADABOOSTR_Fe, X_train, Y_train, cv=kfold, scoring=
    scoring)
msg = '%f (%f)' % (cv_results.mean(), cv_results.std())

scaler = StandardScaler().fit(X_train)
rescaledX = scaler.transform(X_train)
base_estimator = [None]
n_estimators = [10,20,30,40,50,60,70,80,90,100]
learning_rate = [10**-3,10**-2,10**-1,10**0,10**1,10**2,10**3]
loss = ['exponential']

param_grid = dict(base_estimator=base_estimator, n_estimators=n_estimators,
    learning_rate=learning_rate,
                  loss=loss)
model = AdaBoostRegressor()
kfold = KFold(n_splits=num_folds, random_state=seed)
grid = GridSearchCV(estimator=model, param_grid=param_grid, scoring=scoring, cv=
    kfold)
grid_result = grid.fit(rescaledX, Y_train)

print('Best: %f using %s' % (grid_result.best_score_, grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score']
```

```
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print('%f (%f) with: %r' % (mean, stdev, param))

from sklearn.metrics import mean_squared_error
scaler = StandardScaler().fit(X_train)
rescaledX = scaler.transform(X_train)
model = AdaBoostRegressor(base_estimator=None,learning_rate=1, loss='exponential',
    n_estimators=20)
model.fit(rescaledX, Y_train)

rescaledValidationX = scaler.transform(X_validation)
predictions = model.predict(rescaledValidationX)
print(mean_squared_error(Y_validation, predictions))


ADABOOST REGRESSION CODE TO Au
import pandas as pd
import numpy as np
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import AdaBoostRegressor
import math

dataset = pd.read_csv('ouro.csv')
print(dataset.shape)
print(dataset.dtypes)

ADABOOSTR_Au = AdaBoostRegressor()

array = dataset.values
X = array[:,0:3]
Y = array[:,3]
validation_size = 0.20
seed = 7
X_train, X_validation, Y_train, Y_validation = train_test_split(X, Y, test_size=
    validation_size, random_state=seed)

num_folds = 10
seed = 7
scoring = 'neg_mean_squared_error'

kfold = KFold(n_splits=num_folds, random_state=seed)
cv_results = cross_val_score(ADABOOSTR_Au, X_train, Y_train, cv=kfold, scoring=
    scoring)
msg = '%f (%f)' % (cv_results.mean(), cv_results.std())

scaler = StandardScaler().fit(X_train)
rescaledX = scaler.transform(X_train)
base_estimator = [None]
n_estimators = [10,20,30,40,50,60,70,80,90,100]
learning_rate = [10**-3,10**-2,10**-1,10**0,10**1,10**2,10**3]
loss = ['exponential']
```

```
param_grid = dict(base_estimator=base_estimator, n_estimators=n_estimators,
    learning_rate=learning_rate,
                  loss=loss)
model = AdaBoostRegressor()
kfold = KFold(n_splits=num_folds, random_state=seed)
grid = GridSearchCV(estimator=model, param_grid=param_grid, scoring=scoring, cv=
    kfold)
grid_result = grid.fit(rescaledX, Y_train)

print('Best: %f using %s' % (grid_result.best_score_, grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print('%f (%f) with: %r' % (mean, stdev, param))

from sklearn.metrics import mean_squared_error
scaler = StandardScaler().fit(X_train)
rescaledX = scaler.transform(X_train)
model = AdaBoostRegressor(base_estimator=None,learning_rate=1000, loss='exponential'
    , n_estimators=10)
model.fit(rescaledX, Y_train)

rescaledValidationX = scaler.transform(X_validation)
predictions = model.predict(rescaledValidationX)
print(mean_squared_error(Y_validation, predictions))


ADABOOST REGRESSION CODE TO Ni
import pandas as pd
import numpy as np
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import AdaBoostRegressor
import math

dataset = pd.read_csv('niquel.csv')
print(dataset.shape)
print(dataset.dtypes)

ADABOOSTR_Ni = AdaBoostRegressor()

array = dataset.values
X = array[:,0:3]
Y = array[:,3]
validation_size = 0.20
seed = 7
X_train, X_validation, Y_train, Y_validation = train_test_split(X, Y, test_size=
    validation_size, random_state=seed)

num_folds = 10
seed = 7
scoring = 'neg_mean_squared_error'
```

```
kfold = KFold(n_splits=num_folds, random_state=seed)
cv_results = cross_val_score(ADABOOSTR_Ni, X_train, Y_train, cv=kfold, scoring=
    scoring)
msg = '%f (%f)' % (cv_results.mean(), cv_results.std())

scaler = StandardScaler().fit(X_train)
rescaledX = scaler.transform(X_train)
base_estimator = [None]
n_estimators = [10,20,30,40,50,60,70,80,90,100]
learning_rate = [10**-3,10**-2,10**-1,10**0,10**1,10**2,10**3]
loss = ['exponential']

param_grid = dict(base_estimator=base_estimator, n_estimators=n_estimators,
    learning_rate=learning_rate,
                  loss=loss)
model = AdaBoostRegressor()
kfold = KFold(n_splits=num_folds, random_state=seed)
grid = GridSearchCV(estimator=model, param_grid=param_grid, scoring=scoring, cv=
    kfold)
grid_result = grid.fit(rescaledX, Y_train)

print('Best: %f using %s' % (grid_result.best_score_, grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print('%f (%f) with: %r' % (mean, stdev, param))

from sklearn.metrics import mean_squared_error
scaler = StandardScaler().fit(X_train)
rescaledX = scaler.transform(X_train)
model = AdaBoostRegressor(base_estimator=None,learning_rate=1000, loss='exponential'
    , n_estimators=10)
model.fit(rescaledX, Y_train)

rescaledValidationX = scaler.transform(X_validation)
predictions = model.predict(rescaledValidationX)
print(mean_squared_error(Y_validation, predictions))


GRADIENT BOOST REGRESSION CODE TO Fe
import pandas as pd
import numpy as np
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import GradientBoostingRegressor
import math

dataset = pd.read_csv('ferro3.csv')
print(dataset.shape)
print(dataset.dtypes)

GBR_Fe = GradientBoostingRegressor()
```

```
array = dataset.values
X = array[:,0:3]
Y = array[:,3]
validation_size = 0.20
seed = 7
X_train, X_validation, Y_train, Y_validation = train_test_split(X, Y, test_size=validation_size, random_state=seed)

num_folds = 10
seed = 7
scoring = 'neg_mean_squared_error'

kfold = KFold(n_splits=num_folds, random_state=seed)
cv_results = cross_val_score(GBR_Fe, X_train, Y_train, cv=kfold, scoring=scoring)
msg = '%f (%f)' % (cv_results.mean(), cv_results.std())

scaler = StandardScaler().fit(X_train)
rescaledX = scaler.transform(X_train)
learning_rate = [0.1,0.2,0.3,0.4,0.5]
n_estimators = [50,60,70,80,90,100,110]
subsample = [0.1,0.2,0.5,1]
min_samples_split = [2,3,4,5,6,7,8]
min_samples_leaf = [1,2,3,4,5,6]
param_grid = dict(learning_rate=learning_rate, n_estimators=n_estimators, subsample=subsample,
                  min_samples_split=min_samples_split, min_samples_leaf=min_samples_leaf)

model = GradientBoostingRegressor()
kfold = KFold(n_splits=num_folds, random_state=seed)
grid = GridSearchCV(estimator=model, param_grid=param_grid, scoring=scoring, cv=kfold)
grid_result = grid.fit(rescaledX, Y_train)

print('Best: %f using %s' % (grid_result.best_score_, grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print('%f (%f) with: %r' % (mean, stdev, param))

from sklearn.metrics import mean_squared_error
scaler = StandardScaler().fit(X_train)
rescaledX = scaler.transform(X_train)
model = GradientBoostingRegressor(learning_rate=0.3, min_samples_leaf=4, min_samples_split=3, n_estimators=90,
                          subsample=1)
model.fit(rescaledX, Y_train)

rescaledValidationX = scaler.transform(X_validation)
predictions = model.predict(rescaledValidationX)
print(mean_squared_error(Y_validation, predictions))


GRADIENT BOOST REGRESSION CODE TO Au
import pandas as pd
```

```
import numpy as np
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import GradientBoostingRegressor
import math

dataset = pd.read_csv('ouro.csv')
print(dataset.shape)
print(dataset.dtypes)

GBR_Au = GradientBoostingRegressor()

array = dataset.values
X = array[:,0:3]
Y = array[:,3]
validation_size = 0.20
seed = 7
X_train, X_validation, Y_train, Y_validation = train_test_split(X, Y, test_size=
    validation_size, random_state=seed)

num_folds = 10
seed = 7
scoring = 'neg_mean_squared_error'

kfold = KFold(n_splits=num_folds, random_state=seed)
cv_results = cross_val_score(GBR_Au, X_train, Y_train, cv=kfold, scoring=scoring)
msg = '%f (%f)' % (cv_results.mean(), cv_results.std())

scaler = StandardScaler().fit(X_train)
rescaledX = scaler.transform(X_train)
learning_rate = [0.1,0.2,0.3,0.4,0.5]
n_estimators = [50,60,70,80,90,100,110]
subsample = [0.1,0.2,0.5,1]
min_samples_split = [2,3,4,5,6,7,8]
min_samples_leaf = [1,2,3,4,5,6]
param_grid = dict(learning_rate=learning_rate, n_estimators=n_estimators, subsample=
    subsample,
                  min_samples_split=min_samples_split, min_samples_leaf=
                      min_samples_leaf)

model = GradientBoostingRegressor()
kfold = KFold(n_splits=num_folds, random_state=seed)
grid = GridSearchCV(estimator=model, param_grid=param_grid, scoring=scoring, cv=
    kfold)
grid_result = grid.fit(rescaledX, Y_train)

print('Best: %f using %s' % (grid_result.best_score_, grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print('%f (%f) with: %r' % (mean, stdev, param))

from sklearn.metrics import mean_squared_error
```

```
scaler = StandardScaler().fit(X_train)
rescaledX = scaler.transform(X_train)
model = GradientBoostingRegressor(learning_rate=0.5, min_samples_leaf=1,
    min_samples_split=5, n_estimators=50,
                          subsample=1)
model.fit(rescaledX, Y_train)

rescaledValidationX = scaler.transform(X_validation)
predictions = model.predict(rescaledValidationX)
print(mean_squared_error(Y_validation, predictions))


GRADIENT BOOST REGRESSION CODE TO Ni
import pandas as pd
import numpy as np
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import GradientBoostingRegressor
import math

dataset = pd.read_csv('niquel.csv')
print(dataset.shape)
print(dataset.dtypes)

GBR_Ni = GradientBoostingRegressor()

array = dataset.values
X = array[:,0:3]
Y = array[:,3]
validation_size = 0.20
seed = 7
X_train, X_validation, Y_train, Y_validation = train_test_split(X, Y, test_size=
    validation_size, random_state=seed)

num_folds = 10
seed = 7
scoring = 'neg_mean_squared_error'

kfold = KFold(n_splits=num_folds, random_state=seed)
cv_results = cross_val_score(GBR_Ni, X_train, Y_train, cv=kfold, scoring=scoring)
msg = '%f (%f)' % (cv_results.mean(), cv_results.std())

scaler = StandardScaler().fit(X_train)
rescaledX = scaler.transform(X_train)
learning_rate = [0.1,0.2,0.3,0.4,0.5]
n_estimators = [50,60,70,80,90,100,110]
subsample = [0.1,0.2,0.5,1]
min_samples_split = [2,3,4,5,6,7,8]
min_samples_leaf = [1,2,3,4,5,6]
param_grid = dict(learning_rate=learning_rate, n_estimators=n_estimators, subsample=
    subsample,
                  min_samples_split=min_samples_split, min_samples_leaf=
                      min_samples_leaf)
```

```
model = GradientBoostingRegressor()
kfold = KFold(n_splits=num_folds, random_state=seed)
grid = GridSearchCV(estimator=model, param_grid=param_grid, scoring=scoring, cv=
    kfold)
grid_result = grid.fit(rescaledX, Y_train)

print('Best: %f using %s' % (grid_result.best_score_, grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print('%f (%f) with: %r' % (mean, stdev, param))

from sklearn.metrics import mean_squared_error
scaler = StandardScaler().fit(X_train)
rescaledX = scaler.transform(X_train)
model = GradientBoostingRegressor(learning_rate=0.5, min_samples_leaf=1,
    min_samples_split=5, n_estimators=110,
                          subsample=1)
model.fit(rescaledX, Y_train)

rescaledValidationX = scaler.transform(X_validation)
predictions = model.predict(rescaledValidationX)
print(mean_squared_error(Y_validation, predictions))


RANDOM FOREST REGRESSION CODE TO Fe
import pandas as pd
import numpy as np
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestRegressor
import math

dataset = pd.read_csv('ferro3.csv')
print(dataset.shape)
print(dataset.dtypes)

RFR_Fe = RandomForestRegressor()

array = dataset.values
X = array[:,0:3]
Y = array[:,3]
validation_size = 0.20
seed = 7
X_train, X_validation, Y_train, Y_validation = train_test_split(X, Y, test_size=
    validation_size, random_state=seed)

num_folds = 10
seed = 7
scoring = 'neg_mean_squared_error'

kfold = KFold(n_splits=num_folds, random_state=seed)
cv_results = cross_val_score(RFR_Fe, X_train, Y_train, cv=kfold, scoring=scoring)
```

```
msg = '%f (%f)' % (cv_results.mean(), cv_results.std())

scaler = StandardScaler().fit(X_train)
rescaledX = scaler.transform(X_train)

n_estimators = [50,60,70,80,90,100,110]
min_samples_split = [2,3,4,5,6,7,8]
min_samples_leaf = [1,2,3,4,5,6]
max_features = ['auto','sqrt','log2']

param_grid = dict(n_estimators=n_estimators, min_samples_split=min_samples_split,
                  min_samples_leaf=min_samples_leaf, max_features=max_features)

model = RandomForestRegressor()
kfold = KFold(n_splits=num_folds, random_state=seed)
grid = GridSearchCV(estimator=model, param_grid=param_grid, scoring=scoring, cv=
    kfold)
grid_result = grid.fit(rescaledX, Y_train)

print('Best: %f using %s' % (grid_result.best_score_, grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print('%f (%f) with: %r' % (mean, stdev, param))

from sklearn.metrics import mean_squared_error
scaler = StandardScaler().fit(X_train)
rescaledX = scaler.transform(X_train)
model = RandomForestRegressor(max_features='auto', min_samples_leaf=1,
    min_samples_split=2, n_estimators=90)

model.fit(rescaledX, Y_train)

rescaledValidationX = scaler.transform(X_validation)
predictions = model.predict(rescaledValidationX)
print(mean_squared_error(Y_validation, predictions))


RANDOM FOREST REGRESSION CODE TO Au
import pandas as pd
import numpy as np
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestRegressor
import math

dataset = pd.read_csv('ouro.csv')
print(dataset.shape)
print(dataset.dtypes)

RFR_Au = RandomForestRegressor()

array = dataset.values
```

```
X = array[:,0:3]
Y = array[:,3]
validation_size = 0.20
seed = 7
X_train, X_validation, Y_train, Y_validation = train_test_split(X, Y, test_size=
    validation_size, random_state=seed)

num_folds = 10
seed = 7
scoring = 'neg_mean_squared_error'

kfold = KFold(n_splits=num_folds, random_state=seed)
cv_results = cross_val_score(RFR_Au, X_train, Y_train, cv=kfold, scoring=scoring)
msg = '%f (%f)' % (cv_results.mean(), cv_results.std())

scaler = StandardScaler().fit(X_train)
rescaledX = scaler.transform(X_train)

n_estimators = [50,60,70,80,90,100,110]
min_samples_split = [2,3,4,5,6,7,8]
min_samples_leaf = [1,2,3,4,5,6]
max_features = ['auto','sqrt','log2']

param_grid = dict(n_estimators=n_estimators, min_samples_split=min_samples_split,
                  min_samples_leaf=min_samples_leaf, max_features=max_features)

model = RandomForestRegressor()
kfold = KFold(n_splits=num_folds, random_state=seed)
grid = GridSearchCV(estimator=model, param_grid=param_grid, scoring=scoring, cv=
    kfold)
grid_result = grid.fit(rescaledX, Y_train)

print('Best: %f using %s' % (grid_result.best_score_, grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print('%f (%f) with: %r' % (mean, stdev, param))

from sklearn.metrics import mean_squared_error
scaler = StandardScaler().fit(X_train)
rescaledX = scaler.transform(X_train)
model = RandomForestRegressor(max_features='sqrt', min_samples_leaf=1,
    min_samples_split=4, n_estimators=100)

model.fit(rescaledX, Y_train)

rescaledValidationX = scaler.transform(X_validation)
predictions = model.predict(rescaledValidationX)
print(mean_squared_error(Y_validation, predictions))


RANDOM FOREST REGRESSION CODE TO Ni
import pandas as pd
import numpy as np
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
```

```
from sklearn.model_selection import KFold
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestRegressor
import math

dataset = pd.read_csv('niquel.csv')
print(dataset.shape)
print(dataset.dtypes)

RFR_Ni = RandomForestRegressor()

array = dataset.values
X = array[:,0:3]
Y = array[:,3]
validation_size = 0.20
seed = 7
X_train, X_validation, Y_train, Y_validation = train_test_split(X, Y, test_size=
    validation_size, random_state=seed)

num_folds = 10
seed = 7
scoring = 'neg_mean_squared_error'

kfold = KFold(n_splits=num_folds, random_state=seed)
cv_results = cross_val_score(RFR_Ni, X_train, Y_train, cv=kfold, scoring=scoring)
msg = '%f (%f)' % (cv_results.mean(), cv_results.std())

scaler = StandardScaler().fit(X_train)
rescaledX = scaler.transform(X_train)

n_estimators = [50,60,70,80,90,100,110]
min_samples_split = [2,3,4,5,6,7,8]
min_samples_leaf = [1,2,3,4,5,6]
max_features = ['auto','sqrt','log2']

param_grid = dict(n_estimators=n_estimators, min_samples_split=min_samples_split,
                  min_samples_leaf=min_samples_leaf, max_features=max_features)

model = RandomForestRegressor()
kfold = KFold(n_splits=num_folds, random_state=seed)
grid = GridSearchCV(estimator=model, param_grid=param_grid, scoring=scoring, cv=
    kfold)
grid_result = grid.fit(rescaledX, Y_train)

print('Best: %f using %s' % (grid_result.best_score_, grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print('%f (%f) with: %r' % (mean, stdev, param))

from sklearn.metrics import mean_squared_error
scaler = StandardScaler().fit(X_train)
rescaledX = scaler.transform(X_train)
model = RandomForestRegressor(max_features='log2', min_samples_leaf=1,
    min_samples_split=2, n_estimators=110)
```

```
model.fit(rescaledX, Y_train)

rescaledValidationX = scaler.transform(X_validation)
predictions = model.predict(rescaledValidationX)
print(mean_squared_error(Y_validation, predictions))


EXTRA TREES REGRESSION CODE TO Fe
import pandas as pd
import numpy as np
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import ExtraTreesRegressor
import math

dataset = pd.read_csv('ferro3.csv')
print(dataset.shape)
print(dataset.dtypes)

ETR_Fe = ExtraTreesRegressor()

array = dataset.values
X = array[:,0:3]
Y = array[:,3]
validation_size = 0.20
seed = 7
X_train, X_validation, Y_train, Y_validation = train_test_split(X, Y, test_size=
    validation_size, random_state=seed)

num_folds = 10
seed = 7
scoring = 'neg_mean_squared_error'

kfold = KFold(n_splits=num_folds, random_state=seed)
cv_results = cross_val_score(ETR_Fe, X_train, Y_train, cv=kfold, scoring=scoring)
msg = '%f (%f)' % (cv_results.mean(), cv_results.std())

scaler = StandardScaler().fit(X_train)
rescaledX = scaler.transform(X_train)

n_estimators = [50,60,70,80,90,100,110]
min_samples_split = [2,3,4,5,6,7,8]
min_samples_leaf = [1,2,3,4,5,6]
max_features = ['auto','sqrt','log2']

param_grid = dict(n_estimators=n_estimators, min_samples_split=min_samples_split,
                  min_samples_leaf=min_samples_leaf, max_features=max_features)

model = ExtraTreesRegressor()
kfold = KFold(n_splits=num_folds, random_state=seed)
grid = GridSearchCV(estimator=model, param_grid=param_grid, scoring=scoring, cv=
    kfold)
grid_result = grid.fit(rescaledX, Y_train)
```

```
print('Best: %f using %s' % (grid_result.best_score_, grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print('%f (%f) with: %r' % (mean, stdev, param))

from sklearn.metrics import mean_squared_error
scaler = StandardScaler().fit(X_train)
rescaledX = scaler.transform(X_train)
model = ExtraTreesRegressor(max_features='auto', min_samples_leaf=1,
    min_samples_split=2, n_estimators=90)

model.fit(rescaledX, Y_train)

rescaledValidationX = scaler.transform(X_validation)
predictions = model.predict(rescaledValidationX)
print(mean_squared_error(Y_validation, predictions))


EXTRA TREES REGRESSION CODE TO Au
import pandas as pd
import numpy as np
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import ExtraTreesRegressor
import math

dataset = pd.read_csv('ouro.csv')
print(dataset.shape)
print(dataset.dtypes)

ETR_Au = ExtraTreesRegressor()

array = dataset.values
X = array[:,0:3]
Y = array[:,3]
validation_size = 0.20
seed = 7
X_train, X_validation, Y_train, Y_validation = train_test_split(X, Y, test_size=
    validation_size, random_state=seed)

num_folds = 10
seed = 7
scoring = 'neg_mean_squared_error'

kfold = KFold(n_splits=num_folds, random_state=seed)
cv_results = cross_val_score(ETR_Au, X_train, Y_train, cv=kfold, scoring=scoring)
msg = '%f (%f)' % (cv_results.mean(), cv_results.std())

scaler = StandardScaler().fit(X_train)
rescaledX = scaler.transform(X_train)

```

```
n_estimators = [50,60,70,80,90,100,110]
min_samples_split = [2,3,4,5,6,7,8]
min_samples_leaf = [1,2,3,4,5,6]
max_features = ['auto','sqrt','log2']

param_grid = dict(n_estimators=n_estimators, min_samples_split=min_samples_split,
                  min_samples_leaf=min_samples_leaf, max_features=max_features)

model = ExtraTreesRegressor()
kfold = KFold(n_splits=num_folds, random_state=seed)
grid = GridSearchCV(estimator=model, param_grid=param_grid, scoring=scoring, cv=
    kfold)
grid_result = grid.fit(rescaledX, Y_train)

print('Best: %f using %s' % (grid_result.best_score_, grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print('%f (%f) with: %r' % (mean, stdev, param))

from sklearn.metrics import mean_squared_error
scaler = StandardScaler().fit(X_train)
rescaledX = scaler.transform(X_train)
model = ExtraTreesRegressor(max_features='auto', min_samples_leaf=1,
    min_samples_split=3, n_estimators=50)

model.fit(rescaledX, Y_train)

rescaledValidationX = scaler.transform(X_validation)
predictions = model.predict(rescaledValidationX)
print(mean_squared_error(Y_validation, predictions))


EXTRA TREES REGRESSION CODE TO Ni
import pandas as pd
import numpy as np
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import ExtraTreesRegressor
import math

dataset = pd.read_csv('niquel.csv')
print(dataset.shape)
print(dataset.dtypes)

ETR_Ni = ExtraTreesRegressor()

array = dataset.values
X = array[:,0:3]
Y = array[:,3]
validation_size = 0.20
seed = 7
X_train, X_validation, Y_train, Y_validation = train_test_split(X, Y, test_size=
```

```
    validation_size, random_state=seed)

num_folds = 10
seed = 7
scoring = 'neg_mean_squared_error'

kfold = KFold(n_splits=num_folds, random_state=seed)
cv_results = cross_val_score(ETR_Ni, X_train, Y_train, cv=kfold, scoring=scoring)
msg = '%f (%f)' % (cv_results.mean(), cv_results.std())

scaler = StandardScaler().fit(X_train)
rescaledX = scaler.transform(X_train)

n_estimators = [50,60,70,80,90,100,110]
min_samples_split = [2,3,4,5,6,7,8]
min_samples_leaf = [1,2,3,4,5,6]
max_features = ['auto','sqrt','log2']

param_grid = dict(n_estimators=n_estimators, min_samples_split=min_samples_split,
                  min_samples_leaf=min_samples_leaf, max_features=max_features)

model = ExtraTreesRegressor()
kfold = KFold(n_splits=num_folds, random_state=seed)
grid = GridSearchCV(estimator=model, param_grid=param_grid, scoring=scoring, cv=
    kfold)
grid_result = grid.fit(rescaledX, Y_train)

print('Best: %f using %s' % (grid_result.best_score_, grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print('%f (%f) with: %r' % (mean, stdev, param))

from sklearn.metrics import mean_squared_error
scaler = StandardScaler().fit(X_train)
rescaledX = scaler.transform(X_train)
model = ExtraTreesRegressor(max_features='auto', min_samples_leaf=1,
    min_samples_split=2, n_estimators=100)

model.fit(rescaledX, Y_train)

rescaledValidationX = scaler.transform(X_validation)
predictions = model.predict(rescaledValidationX)
print(mean_squared_error(Y_validation, predictions))
```